ON CLUSTERING IMAGES OF OBJECTS

BY

JONGWOO LIM

B.S., Seoul National University, 1997
M.S., University of Illinois at Urbana-Champaign, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

Abstract

The images of an object may look very different under different illumination conditions or viewing directions. This thesis considers the problem of clustering images of various objects into disjoint subsets according to object identity.

The clustering problem has been studied for decades and currently it is one of the most active research fields in machine learning. Compared to other data types considered in the clustering research, images are practically very imporant, but they are very difficult to be handled due to lack of understandings on their intrinsic properties.

First, the hypergraph partitioning problem, i.e., the problem of clustering in domains where the orders of affinity relations are higher than pairwise, is addressed, and a new two-step algorithm for solving this problem is proposed. The proposed algorithm first builds a graph approximation of the hypergraph with a novel approximation scheme, then a spectral graph partitioning algorithm is applied to generate the final clustering result. In addition to the thorough experiments, a theoretical analysis relates the proposed algorithm to existing hypergraph partitioning algorithms and presents the reasons for its superior performance.

Given powerful general clustering algorithms, the key step for solving the image clustering problem is to design an effective similarity measure between images. In this thesis, according to the conditions under which images are taken, three different similarity measures are presented. When the relative position between objects and the camera is fixed but the illumination in the environment changes, the conic affinity and the gradient affinity gives excellent clustering result. For each image in the clustering dataset, the conic affinity assigns non-negative weights to other images in its neighborhood, which represents the likelihood of being from the same object. The

weights represent the global geometric relations among images in the image space. In contrast, the gradient affinity works in a pairwise manner. It directly compares two images based on the probabilistic distribution of the image gradient vectors of an object under different illumination.

Dealing with viewing direction changes is harder since these images do not have a simple global structure in the image space. Instead of considering global relation between images, the local linear structure between nearby images is exploited to determine the closeness between images. By considering affine warps of the images, small 3D pose variation can be handled robustly, which enhances the clustering result over pose variaition.

When the lighting and pose variation occurs simultaneously, we do not have an algorithm that presents reasonably good clustering performance yet. A few candidate algorithms are tested extensively and the result are given in this thesis. Finding an effective similarity measure for both changes still remains as an open problem.

Table of Contents

Chapter 1

Introduction

The goal of clustering is to partition a data set into several disjoint subsets in a manner that elements in a subset are more similar to each other than to elements in other subsets. Humans can perform this job relatively easily based on their knowledge and reasoning, but to automate this has proven to be non-trivial. Moreover the recent explosive increase in the amount of stored data makes it almost impossible to organize and categorize all data by human operators. Thus it is preferable for clustering to be performed with no or minimal human guidance. Finding good (or 'optimal') clustering without any supervision is attractive, since it provides a systematic way to discover the underlying structure in the given data set autonomously.

The clustering problem is also practically important in many areas like computer vision, machine learning, statistics, database, data mining, psychology, VLSI CAD, etc. In computer vision, clustering techniques have been applied to segmenting images or videos [25–27, 30], categorizing a large collection of images [2, 9, 29, 42, 59, 83], establishing correspondences of trajectories between multiple video sequences [35, 73], grouping 2D points according to some geometric model (e.g., Hough transformation), etc. Also it has been widely used and developed in many other areas: grouping relevant data and extracting information from groups in a large database, grouping electronic units into several modules so that the connections between modules are minimized, etc.

However it is not always possible to define the 'best' clustering on a dataset. One dataset may be able to be clustered in several different ways which are all good and natural to human. Kleinberg showed that there does not exist any clustering algorithm which satisfies all three simple and de-

sirable criteria: scale-invariance (the clustering algorithm gives same clustering when the distance values between items are uniformly scaled), richness (the clustering algorithm can generate all possible clusterings of a given dataset), and consistency (the clustering algorithm generates the same clustering result when we shrink the distances within a cluster and expand the distances between clusters) [54]. Besides the existence of multiple plausible clustering goals, the other difficulty in designing a clustering algorithm is that it is hard to define a measure of clustering quality in a quantitative way. In other words, it is hard to translate the qualitative clustering criterion that people have in mind into a mathematical equation of the cluster assignment and each data's properties (of course, many algorithms do use such a criteria, but it may not be satisfactory).

Many researchers have studied general clustering algorithms dealing with points in a high-dimensional space or an affinity graph of data elements, assuming that each data item can be represented as a point in some data space, or assuming the existence of a function that computes the similarity between a pair of data points. If one follow this abstraction scheme, the design of a clustering algorithm involves two issues:

1. how to measure the similarity/distance between data, and

2. how to tell one specific clustering is more favorable than others.

The first issue concerns the similarity between data elements. Given a data set, a (pairwise) similarity measure can be defined as a function which takes a pair of data elements and returns a positive real number, the similarity between the elements. It obviously depends on the type or properties of the data and the objective of clustering. For points in a high dimensional space, the Euclidean distance is the most natural and popular distance measure. As an easy and straightforward example, in perceptual grouping problems (Figure 1.1), the natural criterion of the closeness of points is the Euclidean distance in the space. However, when data items have fields that cannot be represented as a real number (e.g., character strings like book titles or author lists, company organization represented as a hierarchical tree structure, etc.), or when treating each data as a point in a dataspace does not reflect the desired similarity relation (e.g., an image can be represented as

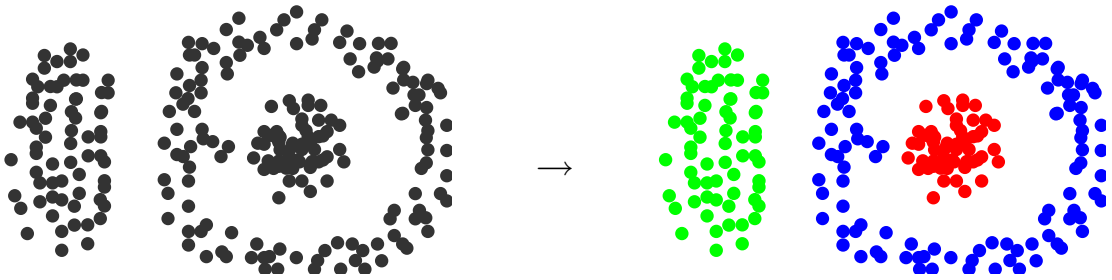


Figure 1.1: Perceptual grouping of 2D points

a high-dimensional real vector, but the Euclidean distance does not capture the semantic relation between images.), a custom similarity measure becomes necessary. Also when there are multiple possible clustering goals (e.g., news articles can be clustered according to their topics, dates, geographic regions, keywords, etc.), corresponding similarity measures need to be designed carefully to reflect the desired goals.

The second measure captures the quality of a clustering. Given a set of data, there are exponentially many ways to partition the set, and this measure must indicate whether a partition we are considering is a good one or a bad one. It is less data- or problem-dependent than the similarity measure, since it works with the computed similarity or distance values between data elements, or sometimes between data elements and a prototype - mean, mediod or something else.

The problem of finding the 'best' assignment of data into a fixed number of bins is known to be an NP-hard problem, and it is unrealistic to try all possible combinations of cluster assignment to find the best one. The best we can find is some approximately optimal cluster assignment of the data set. There are two popular approaches to getting an approximate answer: model-based and graph-based. Model based algorithms maintain and evolve a given number of 'cluster representatives' for the given dataset. Each element is assigned to one of the clusters determined by its distance to the cluster representative, then the cluster representative is adjusted according to the current menbers. This process is iterated until there is no membership changes and no representative updates. k-means , k-median and some algorithms that represent clusters with a probability distribution function and use the EM algorithm fall in this category.

In contrast, graph-based algorithms do not use cluster representatives, nor do they measure the distance from each data to the representatives. As input, they take (all) pairwise similarities between data elements, and they seek a 'good' partition of the graph defined by the data elements (vertices) and the similarities (edge weights). The goodness of a partitioning is measured by the amount of 'distortion' which is induced by cutting edges of the graph to make a partition. There are two popular distortion measures, which are called *min-cut* and *normalized cut* (Ncut). In finding the partition which minimizes those distortions, the algorithms based on the spectral analysis of a graph are the most popular and well-studied algorithms so far [18, 61, 84].

The spectral clustering algorithm is one of the most popular and powerful clustering algorithms. The inputs to the algorithm are all pairwise similarities between elements (represented as an affinity matrix) and the desired number of clusters, and then the algorithm computes the eigen-vectors of the normalized affinity matrix and assigns data items into clusters based on the corresponding eigen-vectors. After the algorithm was introduced, its link to the graph partitioning were discovered [5, 41], and further integration with other algorithms (e.g., kernel k-means algorithm) is being pursued [21]. In the next chapter, we will discuss the spectral clustering algorithm, the normalized cut algorithm and their relation (equivalence) in detail, since we will use the spectral clustering algorithm as the clustering engine for the image clustering task.

1.1 Multi-adic Clustering

Most previous work in clustering deals with pairwise affinity of data. However in many cases, the similarity among data cannot be determined in a pairwise manner, but only with more than two data elements. For example, consider the problem of grouping a set of points that are sampled from a few lines (Figure 1.2.A) according to their original membership. To determine whether or not a collection of data points are from one line, we need at least 3 points to test their fitness to a line (Figure 1.2.B) since any two points trivially form a line.

Such a multi-adic similarity can be considered as a hyperedge in a hypergraph, and then the

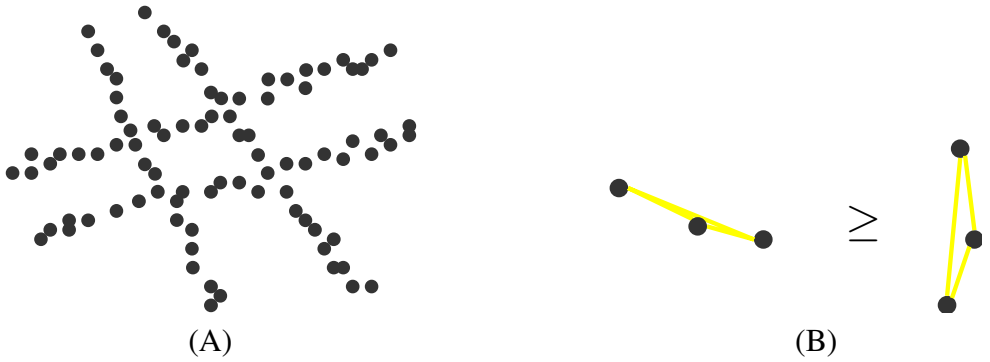


Figure 1.2: Clustering points into lines. (A) Points from four intersecting lines. (B) The points in the left side is more similar to a line than the ones on the right.

clustering problem becomes a hypergraph partitioning problem. A hypergraph is a generalization of a graph, whose hyperedges join more than two vertices, whereas graph edges join two vertices. Just like the graph edges, a hyperedge can have an associated weight.

Once the similarity measure for more than two elements is determined, there are two choices to form clusters: to work directly with the estimated multi-adic similarities, or to approximate them with pairwise similarities and then work with the approximate graph. Both approaches have pros and cons. Usually the hypergraph is more difficult to handle than the graph, inherently because of huge difference in the degrees of freedom of hypergraphs compared to graphs, and historically because of the paucity of theoretically well-studied algorithms for hypergraphs. The approximation approach also has some problems. Obtaining a good approximation may not be always possible, and some information in the original hypergraph is inevitably lost during the process of approximation. However it is not unreasonable to hope that if the original hypergraph is clusterable, there will be a good graph approximation for the clustering purpose. We chose the approximation approach since we have good tools for graph partitioning (and also we have a good understanding for them).

Thus we develop an effective approximation algorithm of a hypergraph, and then clusters are obtained by a graph partitioning algorithm using the resulting graph. Compared to the existing algorithm (Clique Expansion) [90], the proposed *Clique Averaging* algorithm preserves more in-

formation in the approximation than when hyperedges are converted to cliques; instead of just assigning the original hyperedge weight to all edges in the clique, the proposed method is based on solving a linear system of weights of hyperedges and graph edges.

1.2 Clustering Images

There is huge variety in types of real world data stored in computers, and new types of data are introduced as computers are used in more and more areas. Images are one of most common and important data type that are stored in and processed by computers these days. They have a very simple structure (two dimensional array of pixel values), but it is not so easy to deal with images based on their content, e.g., recognizing what is in the images, how many objects are there, and so on. From the clustering point of view, we need to pick a similarity measure for pairs of images cleverly to obtain good clustering results. We design our image clustering algorithm using affinities which are appropriate under certain assumptions on the imaging conditions.

The goal in this thesis is to cluster images of objects according to their identities. Once the similarity measure between a pair of images is defined, we can compute all pairwise similarities, and then use the spectral clustering algorithm to generate a clustering result. The key question in solving the image clustering problem is how accurately the similarity measure reflects the actual grouping relation.

Image clustering is a very difficult problem since the appearance of an object may look drastically different for various reasons including illumination variations, viewing direction changes, and intrinsic deformations (e.g., facial expression changes, articulations and bending). In computer vision, 'viewing condition' typically refer to the relative orientation between the camera and the object (i.e., pose) and the illumination condition under which the image is acquired. These extrinsic conditions are usually the most frequent and important cause of appearance changes of objects in images. We develop simple and effective affinity measures over these two variations.

Often there can be multiple 'valid' clusterings of a dataset depending on the desired granularity

of clusters. For example, if the data set contains images of faces and cars, one may want to cluster them into two groups according to their 'classes' (faces and cars), or into many groups of 'instances' (e.g., person A, person B, person C, car X and car Y). Usually when the data set contains only one object class (e.g., face), the clustering goal is to group by identity [42, 59], but when the dataset contains many different classes of objects, the goal is to group them into object categories [57, 83] (then further processing may find individual object instances). Between these two extremes there exists a hierarchy of category of increasing generalilty in many clustering problems (e.g., vehicle, car, sedan, 2003 Honda Accord, etc.). In this work, we study how the 3D surface geometry and viewing conditions affect the resulting image, and how this can be used in designing similarity measures for clustering. Therefore the clustering criterion in this thesis is focused on finding the individual instances, not on finding a general metric which groups objects from the same class together.

1.2.1 Clustering Images Under Different Illumination Conditions

First we consider the problem of varying illumination. Studies on illumination have shown that images of the same object may look different under different lighting conditions [1], while different objects may appear similar under some illumination conditions [46].

Consider the images shown on the left pane in Figure 1.3. There are two natural ways to group these images: we can cluster them by illumination condition or identity (aligned in horizontally or vertically in the right pane respectively). To cluster images according to their illumination conditions, the fact that the shadow formation is more or less the same can be exploited directly by computing some statistics among pixels. Numerous algorithms for estimating lighting direction have been proposed in the literature, e.g., [62, 81, 89], and undoubtedly many of these algorithm can be applied with few modifications to clustering according to lighting conditions (e.g., k-means algorithm using Euclidean distance in the image space would be the simplest technique, and is successful for this dataset).

On the other hand, clustering by identity is considerably more difficult as the appearance of an

Figure 1.3: Image clustering over illumination variations.

object may vary dramatically as the lighting changes. Prior work on face recognition has shown that the appearance variation of the same person under different lighting condition is almost always larger than the appearance variation of different people under the same lighting conditions [1].

A first glance at the images in Figure 1.3 or standard face image databases (Figure 4.5) may suggest that it is a daunting task to develop an *unsupervised* clustering algorithm to group these images based on *identity*. The feature-based or part-based approaches will be challenged since it is quite difficult to detect features or patches reliably and consistently under large illumination variations. However, in recent few years, there have been many structural results concerning illumination effects on images of 3D objects, e.g., [8, 10, 17, 65]. This work benefits greatly from those results in devising the affinity measures, and we present two very effective affinity measures for clustering unlabeled images of 3D objects acquired at fixed pose under varying illumination conditions.

The *conic affinity measure* is developed from the fact that images of an object under arbitrary lighting form a convex cone in the image space [10]. We can infer from the cone property that one image in the illumination cone can be expressed as a convex combination of other images

surrounding the image in the same cone (e.g., generators of the cone). However, in the clustering problem, we do not have any prior knowledge on the cones of each objects (generators of the cone or subspaces close to the cone). Thus we use all images in the given data set and find out the most likely conic relations between images, finding the closest non-negative linear combination of other images to each image. The estimated coefficients represents how close the images are in the image space in the conic sense: images with large coefficient values are close to the target image. Thus the coefficients can be used as their similarity measures to the target image.

Another similarity measure, called the *gradient affinity measure*, is related to the statistical study of gradient images of a Lambertian object [17]. Suppose that we have two images, and have to decide whether they are from one object under different lightings or two different objects. It is shown that there is no 'illumination invariant' for the images of a Lambertian object; this arise because there always exists a surface and two lightings which could have generated the two given images. However, it is also shown that the gradient magnitudes and directions at each pixel location in the two images have a strong probabilistic relation. In other words, the illumination condition can be arbitrary, but the gradient at each pixel is strongly related to the curvature and albedo change of the point on the 3D surface. The gradient affinity makes use of this observation to give an estimate of the similarity between two images.

We show that both affinity measures give very good clustering results on various datasets by showing experimental results and comparisons to other clustering algorithms.

1.2.2 Clustering Images Over Pose Variations

We also consider the problem of clustering images of objects seen from different viewpoints. A traditional computer vision approach to solve this problem would most likely include some kind of image feature extraction, e.g., texture, shape, filter bank outputs, etc. [29, 70]. The underlying assumption is that some (local) image properties of a 3D object are stable over a wide range of viewing conditions, and their correspondences between images can be established. The drawbacks of such an approach are that it is usually difficult to extract these features consistently when viewing

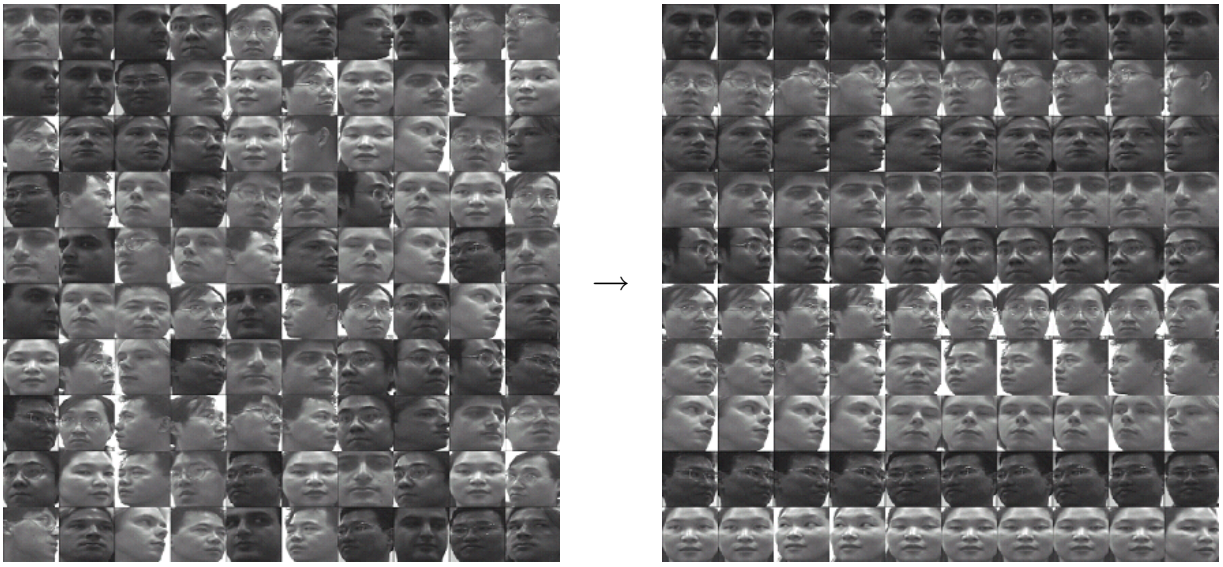


Figure 1.4: Image clustering over pose variations.

direction changes, and also it is hard to associate the detected features between images to evaluate their similarity.

Appearance-based approaches e.g., [9, 25] offer a different strategy for tackling the clustering problem. They start with an understanding of how the images of an object vary under different viewing directions in the image space. For this type of algorithms, image feature extraction no longer plays a significant role. Instead, it is the geometric relations among images in the image space that is the focus of attention. The central geometric concept in appearance-based methods is the idea of an appearance manifold introduced by Murase and Nayar [60].

When estimating the 'distance' between points on manifolds, the Euclidean distance in the space will not give a correct measure since it completely ignores the geodesic relation between points. In addition to the geometric shape of each manifold, it is also important to consider the distance between different manifolds. In terms of image clustering, the similarity measure between two images must represent the likelihood of belonging to the same cluster. For example, two images of different objects may look very similar (close in the image space) and two images of the same object may look very different (far in the image space) depending on viewing directions.

Hence, we only consider the distances among image points located close to each other in the

image space, assuming that it is more likely that those points belong to the same manifold, i.e., there are more points from the same manifold than different manifolds in the neighborhood of the point. The proposed measure, *local linear structure* (LLS), computes the non-negative linear approximation of an image using its neighbor images, and the coefficients will be interpreted as a similarity measure to the image. Even when some nearby images are from different manifolds, the local linear structure measure will assign appropriately small weights to them, and these small weights will be 'intelligently' ignored by the subsequent steps, since the estimated similarities to other images in the same cluster must be larger than these 'mistakes'. Here it is assumed that the viewing direction changes slowly (equivalently, the sampling from the manifold is dense enough), so that the proposed local approximation approach remains valid.

We also exploit the observation that image variations caused by small 3D motions of an object can be well modeled with a 2D affine warp, and more accurate affinities can be achieved by ignoring those variations. As its geometric interpretation, the concept of quotient space is adopted to improve the clustering result.

1.3 Thesis Overview

This thesis is organized as follows. Chapter 2 gives a brief review of spectral clustering, the normalized cut graph partitioning algorithm and various previous works about image clustering. Then multi-adic clustering (hypergraph partitioning) is discussed in the Chapter 3, and a few preliminary experimental results are presented. This work was presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2005 [2]. Chapters 4 and 5 cover the topics clustering and affinity measures for images under different lighting conditions and different viewing directions respectively. The research on the illumination variation has been published in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2003 [42], and the research on pose variation was published in the Eighth European Conference on Computer Vision (ECCV 2004) [59]. Chapter 6 shows the extensions of the local linear structure method used in Chapter 5 to handle

simultaneous light and pose changes. Finally Chapter 7 contains a discussion about the general image clustering problem and future research directions.

Chapter 2

Related Work

To cluster items in a data set, we must be able to measure the similarity between pairs or groups of items. The clustering problem is one of most popular and important topics in unsupervised learning area, and there has been huge amount of work in this field spanning a few decades. Jain et al. [47] reviewed various clustering algorithms and applications of the clustering problem. They presented a few criteria for categorizing clustering algorithms: hierarchical vs. partitional, agglomerative vs. divisive, monothetic vs. polythetic (whether the features are used one-by-one or all at the same time), hard vs. fuzzy class assignment, deterministic vs. stochastic, and/or incremental vs. non-incremental. Other components like similarity measure or representation of clusters are also very important characteristics of clustering algorithms.

The most straightforward way of performing a clustering task is to treat each data item as a point in an abstract metric data space, measure the Euclidean distance between items or between an item and a class representative in that space, and group items or determine each item's membership according to the measured distances. Many model-based clustering algorithms like k-means and EM-based algorithms use this approach.

Probabilistic interpretation of these algorithms is well studied in Duda et. al.'s book [22], and the book also gives good introduction to hierarchical clustering algorithms and other unsupervised learning techniques. Ripley's book [68] is another good introductory book for supervised and unsupervised learning techniques, including statistical decision theory, linear/non-linear discriminants and artificial neural networks for classification problems, and principal component analysis,

multidimensional scaling and various clustering techniques.

The k-means algorithm has been used widely due to its simplicity, speed and performance when data are well clustered in a vector space. The k-means algorithm starts with a set of centers that represent an initial guess for the center of each cluster. It assigns each datum to a cluster according to the distance to the centers, and then updates the location of each center to the mean position of its members. It iterates until there is no change of membership. It is known that in a finite number of trials, the k-means algorithm converges to one of local minima of the cost function

$$E_{kmeans}(\{\mu_i\}; \mathcal{D}) \; = \; \sum_{x \in \mathcal{D}} \min_i \|x - \mu_i\|^2 \tag{2.1}$$

where μ_i's are the centers and $\mathcal{D}$ is the set of data, therefore it is guaranteed that it will not iterate indefinitely.

Since the k-means algorithm finds the solution in an iterative manner and it converges to a local minimum, it is important to provide good initial centers to the algorithm. The most common approach is to run many trials with random initializations and find the best among them, but also a lot of heuristic methods for initialization have been developed to find probably better (or avoid worse) initializations. In this work, the k-means algorithm is used as the last step in the spectral clustering process to generate the final cluster assignment of the embedded points. The standard k-means algorithm with the following heuristic for initialization is used for this purpose: an initial center is picked from the data set randomly, and next centers are iteratively chosen as the farthest point from previously chosen centers, this continues until all centers are determined. This heuristic prevents initializations with centers located close to each other, and thus accelerates the convergence. As shown in [43], this initial clustering is no worse than twice the optimal according to the criteria in Equation 2.1.

However, depending on the types of data, it may not be possible to define a space in which the dissimilarity between elements is measured as a distance, and this is essential for the model-based algorithms. For example, defining a metric for the names of authors or keywords of news articles is

very difficult since a pair will only match or mismatch each other, and there is no natural 'distance' among them. This has been the most significant intrinsic limitation of the model based algorithms.

We assume that the similarity between pairs of elements can be calculated by 'some' unknown function F, then the task left to us is grouping elements according to their similarity. Let us denote the set of data items as $\mathcal{D}$ and a similarity measure or cost function as $F : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$, whose domain is every pair of elements and which returns the similarity between the pair. Also the i-th item in the data set is denoted as i for notational convenience. The core properties of the measure F is non-negativity and symmetricity: $F(i,j) \geq 0$, $F(i,j) = F(j,i)$, $\forall i,j \in \mathcal{D}$. One may also include the triangle inequality property, $F(i,j) \leq F(i,k) + F(k,j)$, but this is not an essential assumption in doing clustering. Now the clustering problem can be formulated as finding a partition Π of the data set $\mathcal{D}$, given the similarity function F, which minimizes a certain cost function $E(\Pi, F)$. Here the partition Π of $\mathcal{D}$ is a set of disjoint subsets $\mathcal{D}_k$'s of $\mathcal{D}$, i.e., $\mathcal{D}_k \cap \mathcal{D}_l = \emptyset$, $\forall k \neq l$ and $\bigcup_k \mathcal{D}_k = \mathcal{D}$. The choice and properties of the cost functions will be discussed in the following section.

For a given data set $\mathcal{D}$, all pairwise similarities of elements in $\mathcal{D}$ can be represented as a symmetric affinity matrix $A \in \mathbb{R}^{n \times n}$, where $A_{ij} = F(i,j)$ and n is the number of elements in $\mathcal{D}$. Then the data set and the affinity matrix can be thought of as a complete graph $G = (\mathcal{D}, A)$, and the original clustering problem becomes equivalent to finding a partition of the graph that minimizes some cost function. Clearly, the clustering problem is very closely related to the graph partitioning problem.

Besides the algorithms using eigenvectors of the graph weight matrix, many graph-based algorithms were studied. The divisive clustering algorithm using the minimum spanning tree (MST) of the graph is introduced by Zahn [87]. In his algorithm, the clustering is achieved by repeatedly breaking the longest edge in the MST of the graph. Another approach for clustering with MST is to break 'inconsistent edges', which are the edges with significantly larger weights than other edges shareing a vertex with it. Also it is possible to group vertices based on the edge-weight histogram [22]. However it is also known that these graph-based algorithms does not minimize a

certain global error function, and they are sensitive to details of data. Toussaint's relative neighborhood graph [80] uses the Delaunay graph (which contains more data than MST) to generate the clustering. However they are mostly used as a classifier substituting the nearest-neighbor classification algorithm, but rarely applied to unsupervised learning problems [48].

2.1 Normalized Graph Cut

The graph partitioning problem is about finding a partition Π of $\mathcal{D}$ given a graph $G(\mathcal{D}, A)$, which minimizes some cost function $E(\Pi; \mathcal{D}, A)$. First we will consider the bi-partitioning problem, i.e., finding $\Pi = \{\mathcal{D}_0, \mathcal{D}_1\}$ where $\mathcal{D}_1 = \mathcal{D} \setminus \mathcal{D}_0$, and then extend it to the multi-subset partition case later in this section.

The cost of a partition represents the amount of distortion induced by removing some edges for partitioning. It can be defined in various ways depending on the desired clustering goals, and here we introduce two of the most well-known cost functions for graph partitioning algorithms. The cut value of a partition is defined as:

$$cut(\mathcal{D}_0, \mathcal{D}_1) \;=\; \sum_{i \in \mathcal{D}_0, j \in \mathcal{D}_1} A_{ij} \;, \quad \mathcal{D}_0 \cap \mathcal{D}_1 = \emptyset$$

This value is the sum of weights of all edges which cross the partition boundary (Figure 2.1.B). The *minimum cut* partition (*min-cut*) of $\mathcal{D}$ is the partition whose cut value is smallest among all possible partitions of $\mathcal{D}$. The problem of finding the *min-cut* partition has been well studied, and there exist an efficient solution for it [85]. The *min-cut* partition works well for some problems, but it is also known that the *min-cut* algorithm tends to generate small sets with only one isolated node in the graph, thus it may fail when there are many outliers in the data set. This is a natural consequence of the definition of the cut criterion, since the sum of weights of edges containing an isolated vertex is usually smaller than edges containing other vertices that are located closely to each other. Figure 2.1.A illustrates the case.

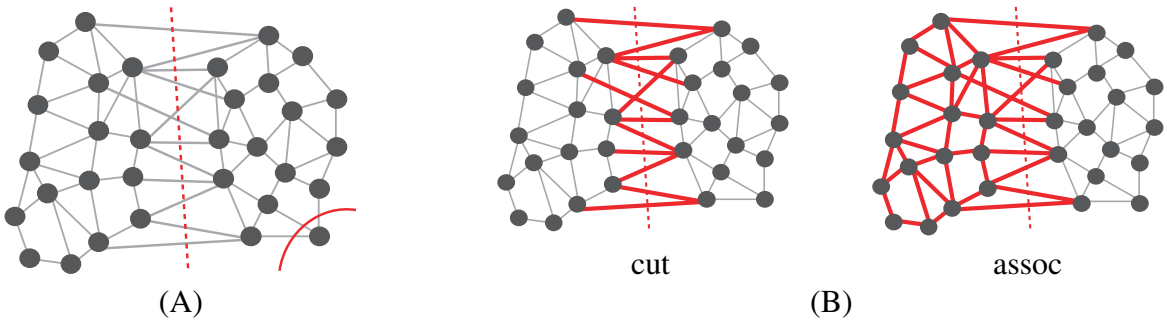


Figure 2.1: *Minimum cut* and *Normalized cut* (A) The cut value is the sum of weights of edges crossing the cut (red dotted line). The *min-cut* algorithm tends to find a set with only one isolated node (red solid line). (B) The *normalized-cut* value is the sum of ratios of each subset's cut value (sum of red edge weights in the left plot) to its $assoc$ value (sum of red edge weights in right). Small groups with few nodes are penalized in the $Ncut$ formulation.

To avoid this problem, Shi and Malik [74] proposed a new measure called the *normalized cut* ($Ncut$, Figure 2.1.B). To define the normalized-cut, they first defined the $assoc$ value of a subgraph in a graph, $\mathcal{A} \subset \mathcal{D}$, as $assoc(\mathcal{A}, \mathcal{D}) = \sum_{i \in \mathcal{A}, j \in \mathcal{D}} A_{ij}$. This value gives the total connection from the subgraph to the graph. Then the *normalized cut* is defined as,

$$Ncut(\mathcal{D}_0, \mathcal{D}_1) \;=\; \frac{cut(\mathcal{D}_0, \mathcal{D}_1)}{assoc(\mathcal{D}_0, \mathcal{D})} + \frac{cut(\mathcal{D}_0, \mathcal{D}_1)}{assoc(\mathcal{D}_1, \mathcal{D})}.$$

The $Ncut$ measures the ratio of edge weights being cut to all edge weights of the subgraph that is made by the partition. One can also define the *normalized association* similarly,

$$Nassoc(\mathcal{D}_0, \mathcal{D}_1) \;=\; \frac{assoc(\mathcal{D}_0, \mathcal{D}_0)}{assoc(\mathcal{D}_0, \mathcal{D})} + \frac{assoc(\mathcal{D}_1, \mathcal{D}_1)}{assoc(\mathcal{D}_1, \mathcal{D})},$$

and interestingly it has been proven that minimizing $Ncut$ is equivalent to maximizing $Nassoc$ [74]. Let L be a normalized Laplacian of the graph $G(\mathcal{D}, A)$, which is defined as

$$L \;=\; D^{-1/2}(D - A)D^{-1/2}$$

where D is an $n \times n$ diagonal matrix whose (i, i)-element is the sum of elements in the i-th row

of A except the diagonal elements in A, i.e., $D_{ii} = \sum_{j \neq i}^{n} A_{ij}$. All diagonal elements of A are set to zero before computing the Laplacian L. The partition is achieved by computing the eigenvector corresponding to the second smallest eigenvalue of the normalized Laplacian L of the graph, and then assigning each vertex i to $\mathcal{D}_0$ if the i-th value of the eigenvector is negative, and assigning it to $\mathcal{D}_1$ otherwise. They showed that this is the partition of $G(\mathcal{D}, A)$ minimizing the normalized cut criterion. The detailed proofs of these results can be found in [74]. The normalized cut algorithm was applied to the image segmentation, tracking and motion segmentation problems, and resulted in good performance [73]. In this approach, solving the eigen-problem is the most computation-intensive component, and so Fowlkes et al. proposed the Nyström method for efficiently computing the eigenvectors in order to handle large data sets [27].

When there exist more than two clusters, one way to recover them is to recursively apply the bi-partitioning algorithm to the partitioned data set. However it is a difficult task to determine which subset in the current partition needs to be further partitioned. Another possibility is to use the eigenvectors corresponding to the eigenvalues less than the second eigenvalue. Many researchers [5, 86, 88] used the eigenvectors associated with the k smallest eigenvalues except the smallest one, where k is the desired number of clusters. The rows of the k eigenvectors are treated as points embedded in $\mathbb{R}^k$, and the distance between points in $\mathbb{R}^k$ represents their similarity. Usually the k-means algorithm is used to cluster them into k groups to generate the final partitioning.

2.2 Spectral Clustering

After Ng et al. [61] introduced an ideal-case analysis of the spectral clustering algorithm, a lot of research has been done to understand its mechanism, its expected performance and its relation to other clustering/partitioning algorithms in related fields. The algorithm is very simple and easy to implement if an efficient eigensolver is available (see Figure 2.2).

Recently Bach and Jordan [5] showed the relation between the spectral clustering algorithm and the normalized graph cut algorithm. They showed that the two problems are exactly equivalent.

1. **Inputs**
   An affinity matrix $A \in \mathbb{R}^{n \times n}$ and the number of clusters k.

2. **Build the Normalized Laplacian Matrix**
   Set $A_{ii} = 0$, and build a diagonal matrix D such that $D_{ii} = \sum_{j=1}^{n} A_{ij}$.
   Construct the matrix $L = D^{-1/2} A D^{-1/2}$.

3. **Compute the eigenvectors**
   Find $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, the k largest eigenvectors of L, and form the matrix
   $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k] \in \mathbb{R}^{n \times k}$.

4. **Build the embedded point set**
   Normalize each of X's rows to have unit length, and treat each row $\mathbf{p}_i$ as a point in $\mathbb{R}^k$,
   $\mathbf{p}_i = [X_{i1}, X_{i2}, \ldots, X_{ik}]^\top / \sqrt{\sum_j X_{ij}^2}$

5. **Cluster the embedded points**
   Cluster the points $\mathbf{p}_i$'s into k clusters via k-means or any other algorithm. Finally assign the original item i to cluster j if and only if $\mathbf{p}_i$ was assigned to cluster j.

Figure 2.2: The spectral clustering algorithm [61].

To obtain the optimal (relaxed) solution, the weighted k-means algorithm must be used instead of k-means algorithm in the original algorithm proposed by Ng et al. For a detailed derivation and proof, refer to their technical report [4].

There have been two different ways to use the affinity matrix when doing the spectral analysis: one uses the normalized affinity matrix ($D^{-1/2} A D^{-1/2}$ or AD^{-1}), and the other uses the original unnormalized affinity matrix (A). Both algorithms have been used in various applications with good empirical results in practice, but it was not clear which approach is correct or works better. Von Luxburg et al. [82] addressed this question as the convergence of the resulting clustering when the amount of data goes to infinity. They showed that the normalized spectral clustering algorithm usually converges, while the unnormalized algorithm only converges under strong additional assumptions, which are not always satisfied.

2.3 Multi-adic Clustering

All clustering or graph partitioning algorithms discussed so far use pairwise affinities or a graph representation. Here we briefly survey the efforts for clustering with multi-adic similarities or partitioning of hypergraphs.

The study of distances defined over sets of size greater than two is not a new enterprise. The literature on n-metrics is devoted to constructing and analyzing distance measures defined over $(n+1)$-tuples. In this notation the usual pairwise metrics are referred to as 1-metrics. The primary focus of this literature is the study of topological and geometrical properties of these generalized measures [20].

While the work on n-metrics is theoretical, a more practical line of work has emerged in the psychometrics community. Starting with the work of Hayashi, who proposed the area of a triangle as the triadic distance between its vertices [39], a number of researchers have developed generalizations of Multidimensional Scaling (MDS) to the case of triadic data. MDS is a technique for embedding pairwise similarity or dissimilarity data in a low dimensional Euclidean space [14]. This embedding is primarily used for the purposes of visualization but it can also be used as a preprocessing step for data analysis methods that require a coordinate representation of their input. In one of the earliest such works, Carroll and Chang developed an algorithm for n-adic MDS using a generalization of SVD to the case of n-dimensional matrices [16]. Subsequently, Cox et al. have proposed an MDS algorithm based on a combination of a Gradient Descent and Isotonic regression [19]. Axiomatic theories of triadic distances have been developed by Joly & LeCalvé and Heiser & Bennani [40, 49]. Joly & LeCalvé propose the use of a least squares procedure for recovering pairwise distances and running classical MDS on them, and, Heiser & Bennani introduce provably convergent algorithms based on iterative majorization that directly solve for the Euclidean embedding.

The most extensive and large scale use of hypergraph partitioning algorithms, however, occurs in the field of VLSI design and synthesis. A typical application involves the partitioning of large

circuits into k equally sized parts in a manner that minimizes the connectivity between the parts. The circuit elements are the vertices of the hypergraph and the *nets* that connect these circuit elements are the hyperedges [3]. The leading tools for partitioning these hypergraphs are based on two phase, multi-level approaches [50]. In the first phase, a hierarchy of hypergraphs is constructed by incrementally collapsing the hyperedges of the original hypergraph according to some measure of homogeneity. The second phase starts with a partitioning of the hypergraph at its coarsest level, and the algorithm works its way down the hierarchy. At each stage the partition at the level above serves as an initialization for a heuristic that refines the partitioning greedily. These heuristics are based on vertex swaps between the various partitions [24, 52]. Although this results in reasonable performance in real world applications, the development of these tools is almost entirely heuristic and very little theoretical development exists that analyzes their performance beyond empirical benchmarks.

The set of tools and methodology for partitioning graphs is much better developed than those for hypergraphs. A case in point is the development of algorithms for solving the max-flow min-cut problem on hypergraphs. While extremely efficient algorithms for the case of graphs have been available for sometime now [33], it is only recently that efficient algorithms that operate directly on hypergraphs have become available [64]. Thus it makes sense to consider methods that construct a graph that approximates the hypergraph and partition it; this partition in turn induces a vertex partitioning on the original hypergraph. In fact, it is possible to construct methods which operate directly on the hypergraph while implicitly working on its graph approximation [32]. The two most commonly used graph approximations are known as *Clique Expansion* and *Star Expansion*. Clique Expansion, as the name implies, expands each hyperedge into a clique on its vertex set and adds the hyperedge weight to all edges in the clique. As hyperedges are converted to cliques, the approximated graph is updated to reflect the information in the hypergraphs, and finally it gives a graph that approximates the hypergraph. Star expansion, instead, introduces a dummy vertex for each hyperedge and puts edges connecting each vertex in the hyperedge to the dummy vertex [44]. The hyperedge weight is distributed equally to the introduced edges. Compared to the Clique

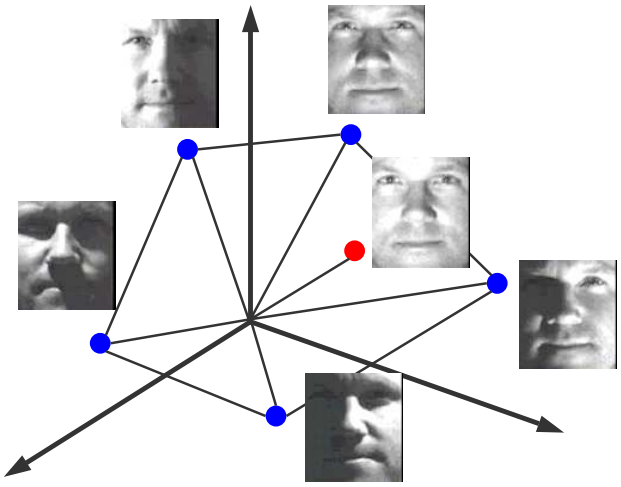


Figure 2.3: Illumination Cone : The images of an object under arbitrary lightings form a convex cone in the image space. An image in the cone can be represented as a convex combination of other images in the cone.

Expansion, Star Expansion creates a much bigger graph in both number of vertices and number of edges. As can be expected, the weights on the edges of the clique and the star determine the approximation and cut properties of the approximating graph. We refer the reader to [37, 45] for further discussions that address this problem.

2.4 Image Clustering

Due to the huge variety of types of data and clustering goals, there cannot be any generic algorithm which computes the similarity between data elements. Therefore the question of how to estimate the similarity between data for a particular application is critical for obtaining good clustering results. We concentrate on the problem of clustering images of objects according to their identities. Clustering images of 3D objects has long been an active field of research in computer vision (See literature review in [9, 11, 12, 15]).

In this section we give a brief overview on various previous works which are directly related to the proposed similarity measures. Belhumeur and Kriegman [10] presented many insightful observations on the characteristics of images under different lightings. It is assumed that the surface

geometry of the object and the relative location and direction between the object and the camera does not change at all. Given a set of these images, they are resized and cropped so that each pixel in the image represents the same point on the 3D object surface. The images can be interpreted as a vector in the image space $\mathbb{R}^N$, where N is the number of pixels in each image. They showed that the images of an object under all possible illumination conditions form a convex cone in the image space, which is natural consequence of the additive property of lights. If the surface is Lambertian and convex (i.e., there is no cast shadow), then all possible images form a convex polyhedral cone (the number of its generators is finite). Further they claimed that the volume of the cone is small (i.e., the cone is skinny), and it can be approximated well by a low-dimensional subspace.

Basri & Jacobs and Ramamoorthi & Hanrahan [8, 65] proposed the use of spherical harmonics to approximate arbitrary lighting on a Lambertian surface with four or nine spherical harmonics coefficients. Lee et al. [56] presented that there exists a set of nine lighting directions which is effective in face recognition or reconstruction of images under various lighting conditions.

Chen et al. [17] proved that there is no 'illumination invariants' in the images of a Lambertian object, by showing that for any two images there always exist a Lambertian surface and two lighting directions which generate the two images. This implies that it is impossible to tell whether two images are taken from a single object or two different objects. However they also showed that the gradient directions at each pixel in the images of an object are strongly correlated, and they suggested a face recognition algorithm based on that probability distribution of image gradients.

Murase and Nayar [60] introduced the concept of 'appearance manifold' to explain the geometric structure of images of an object viewed from various directions in the image space. Basri et al. [9] did pioneering work on image clustering without relying on the extracted feature points. They mainly make use of the geometric structure of points in the image space, by restricting image comparisons in a local area around the target image, and using linear approximation of neighboring image points. Fitzgibbon and Zisserman [25] presented a clustering algorithm for frames in a video sequence. They advocated the use of 2D affine warps to handle small 3D motions between frames.

Chapter 3

Hypergraph Clustering

With a few notable exceptions, formulations of the clustering problem and the proposed algorithms for solving them are based on the assumption that a pairwise (or dyadic) measure of distance between data points is available. A common measure for data points lying in a vector space is the Euclidean distance. The use of a pairwise measure is characteristic of central clustering methods like k-means and k-medoids, as well as pairwise clustering methods based on graph partitioning, semi-definite programming, etc. [30, 51, 53, 74].

It is not always the case, however, that there exists a similarity measure between pairs of data points. For some clustering problems, one may need to consider three or more data points together to determine if they belong to the same cluster. Consider a k-lines algorithm which clusters data points in a d-dimensional vector space into k clusters where elements in each cluster are well approximated by a line. As every pair of data-points trivially defines a line, there does not exist a useful similarity measure between pairs of points for this problem. Yet there do exist useful measures for triplets of points which indicate how close the three points are to being collinear.

Weighted undirected graphs serve as a combinatorial representation for datasets containing pairwise relationships. For this reason, clustering algorithms are also frequently referred to as graph partitioning algorithms. The corresponding representation for datasets with higher order relationships is a hypergraph. Like a weighted graph, a weighted hypergraph is defined as a set of vertices and a set of weighted hyperedges. Each weighted hyperedge can now be an arbitrary subset of the vertices and has a scalar weight associated with it.

The focus of this chapter is the largely neglected but fundamental problem of clustering data on the basis of triadic and higher-order affinities. We introduce a general purpose hypergraph partitioning algorithm, based on a novel graph approximation scheme we call *Clique Averaging*, and show that with an appropriate similarity measure, this generic clustering algorithm can be applied to a number of clustering problems that arise in computer vision.

As an example of the kind of problems we are interested in, consider the problem of clustering a collection of images of different objects, each of which is imaged in the same pose, but under different lighting conditions. Jacobs et al. have shown that for any two images, there exists a Lambertian surface with spatially varying albedo and a pair of light source directions that could have produced the two images [46]. Hence, there is no function of a pair of images that returns zero when the images depict the same object (but under differing lighting), yet returns a non-zero value when the images are depicting different objects.[1] Furthermore, it is well known that the set of images of a Lambertian surface under arbitrary lighting (without shadowing) lies on a 3-D linear subspace in the image space [10]. As three images span this subspace, one can cluster the images of objects by considering tuples of at least four images, and forming an affinity measure on these tuples.

As another example, consider the problem of partitioning a set of point correspondences into clusters that are related by the same motion model (Figure 3.1). The usual approaches are based on

1. doing a greedy set covering using RANSAC [79],

2. performing a Hough Transform [7], or

3. performing pose clustering in the space of the model parameters [71].

There are fundamental problems with each of these approaches. RANSAC was designed for detecting a single model in the presence of a noise, and as we will show it does not scale well to the

[1] However, Chen et al. [17] gave a probabilistic study on the relation of gradient images of the same object under different lightings, which is discussed in Chapter 2 and used in Chapter 4 as the gradient affinity.

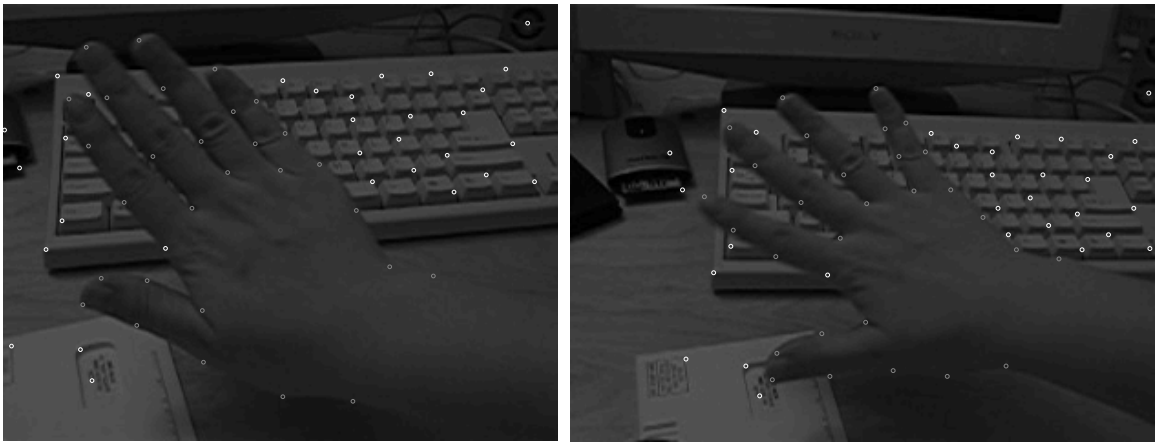


Figure 3.1: Clustering point correspondences in two images. Given the set of point correspondences in two images, the goal is to cluster the correspondences according to the 'coherent' motions. We used the affine motion model, and it can cluster successfully although there is some non-affine deformation (hand), since many 'piecewise affine' correspondences are merged to form a larger set.

case of multiple overlapping models. Approaches based on a generalized Hough Transform require a bounded finite parameterization of the model. This is not a trivial problem, and even if one is available, the Hough transform for anything but the simplest problems requires huge amounts of memory. Clustering in the space of model parameters, while conceptually attractive, may not be tractable. The problem is that to perform clustering in model space, one has to be able to define a metric or a measure of similarity. This is neither simple nor always possible to do in a consistent manner. The parameter spaces for most models are non-linear manifolds without a global metric. In contrast, the fitting error of a set of points to a model is a natural and easily available measure of disassociation, without any limitations on the geometric structure of the parameter space of the model.

3.1 Theory

In this section, we describe our proposed hypergraph partitioning algorithm. It is a two-step procedure. In the first step we construct a weighted graph that approximates the hypergraph. This

approximation is based on a novel algorithm that we call *Clique Averaging*. In the second step we use a spectral clustering algorithm based on the normalized Laplacian of the graph to partition its vertex set. As the second step of this algorithm is well known, we will mainly focus on the development and properties of Clique Averaging. As part of our analysis we will also show the relation of Clique Averaging to Clique Expansion, which is a commonly used graph approximation in the VLSI CAD literature. Finally as a consequence of this relation we are able to show the relation between the algorithm proposed by Gibson et al. [32], and our algorithm. We begin with some notation.

A weighted undirected hypergraph H is a pair (V, h). Here V is the set of vertices of H, and subsets of V are known as hyperedges. The function h associates non-negative weights with each hyperedge. In the special case when the cardinality of the hyperedges is 2, H is a weighted undirected graph and the hyperedges are the same as ordinary graph edges. We use $G = (V, g)$ to denote a weighted undirected graph defined over the same set of vertices V with the weighting function denoted by g. We will assume that the number of vertices in the hypergraph is n, i.e., $|V| = n$. While general hypergraphs can have hyperedges of varying cardinality, and the algorithms we present will work on hypergraphs with arbitrarily sized hyperedges, we will for reasons of notational simplicity assume that all hyperedges are of a fixed known size $k \geq 2$. The two weighting functions h and g can then formally be described as

$$\begin{aligned} h &: V^k \rightarrow \mathbb{R}^+ \\ g &: V^2 \rightarrow \mathbb{R}^+. \end{aligned}$$

As the hypergraphs we are dealing with are undirected, the functions h and g are symmetric in their arguments, i.e., their value remains the same if the order of the arguments is arbitrarily permuted. Finally we will use the symbols d_k to denote the vector of hyperedge weights obtained from H by ordering the hyperedges in lexicographic order based on their vertex sets. The vector d_2 denotes the corresponding lexicographically ordered weight vector for the graph G.

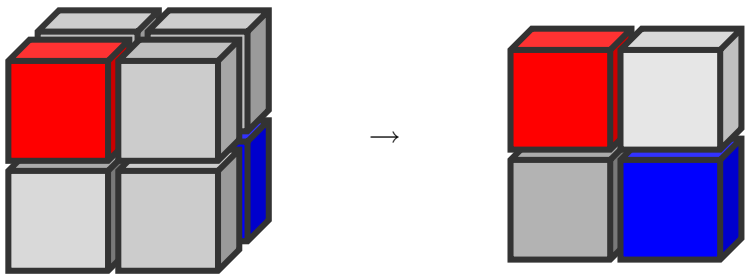


Figure 3.2: Approximating a clusterable hypergraph by a graph. If the hypergraph is clusterable (contains strong block-diagonal structures), it is likely that it can be well approximated by a graph.

In the above notation, the problem of approximating the hypergraph with a graph can now be restated as the problem of approximating the weighting function h with the weighting function g. But before we introduce our approximation scheme, it is instructive to consider the feasibility of such an approximation in a purely combinatorial sense.

For a complete weighted hypergraph of order k defined on n vertices, the weighting function h can take on $\binom{n}{k}$ different values. For a complete graph on the same number of vertices, the number of degrees of freedom is only $\binom{n}{2}$. Even for moderately sized n and k, the number of degrees of freedom for a graph is a tiny fraction of that of a hypergraph.

Thus it is not reasonable to expect any graph approximation with the same number of vertices to do a good job of approximating every possible hypergraph. However we are not interested in approximating all possible hypergraphs. For a dataset to be *clusterable* the weighting function should be indicative of the cluster structure in the data. For example in the case of bi-partitioning a hypergraph, the ideal hypergraph would consist of two completely connected components. The set of hypergraphs of this type is much smaller than $\binom{n}{k}$. In fact it is only $O(n^2)$ and is easily represented as graphs containing two completely connected components on the corresponding vertices. But real data is noisy, and the corresponding hyperedge weights reflect that; however, for data that can be divided into well separated partitions, one would expect that the corresponding hypergraph is close to a hypergraph containing two densely connected components and thus is amenable to approximation with a weighted graph (see Figure 3.2).

3.1.1 Clique Averaging

We are now ready to introduce our hypergraph approximation scheme. Our construction and analysis of the approximation will be based on considering graph approximations of a single hyperedge z. The extension to the whole hypergraph is then a matter of linear superposition. We begin by revisiting the observation that the value of the weighting function $h(z)$ is independent of the order in which we consider the vertices in z. In light of this, when considering the variety of graphs that can be associated with the hyperedge z, the only graph structure that satisfies the requirements of symmetry is the k-clique on z. A k-clique is a completely connected graph on k vertices. Thus the task of approximating $h(z)$ boils down to assigning weights to the edges in the k-clique associated with z.

As was mentioned earlier, the most widely used such approximation scheme is Clique Expansion [90] and is based on the assumption that every edge in the clique associated with z has edge weight equal to $h(z)$. Formally

$$h(z) \;=\; g(v_i, v_j), \qquad \forall v_i, v_j \in z$$

Collecting the above set of equations over all hyperedges results in an over-determined linear system consisting of $\binom{k}{2}\binom{n}{k}$ equations. This system has a simple least squares solution given by

$$g(v_i, v_j) \;=\; \frac{1}{\mu(n,k)} \left(\sum_{v_i, v_j \in z} h(z) \right).$$

Here $\mu(n,k) = \binom{n-2}{k-2}$ is the number of hyperedges that contain a particular pair of vertices. Thus the weight on an edge is the arithmetic mean of the weights of all the hyperedges that contain both of its vertices. Other choices for $\mu(n,k)$ are also possible and will amount to different weighting schemes when working with hyperedges of varying sizes. The optimal choice of weighting in Clique Expansion when combining information across hyperedges is an area of research in itself [37, 45].

The relationship between a hyperedge and the edge weights in its clique in the above approach was the simplest possible, where we assumed that the hyperedge weight and the edge weights are equal to each other. In an attempt to make this relationship richer, we take a generative view of the problem, i.e., let us assume that there exists a $\binom{k}{2}$-ary function F such that, given the edge weights on a k-clique, it calculates the corresponding hyperedge weight. Formally

$$h(z) \;=\; F(g(v_1, v_2), \ldots, g(v_i, v_j), \ldots, g(v_{k-1}, v_k)). \tag{3.1}$$

Now given a particular generative model F and a hypergraph H, the hypergraph approximation problem can then be stated as the problem of solving for those values of the graph edge weights $g(v_i, v_j)$ that satisfy the above equation over all hyperedges simultaneously. Of course how well the graph G captures the structure of hypergraph H is now a function of F. So what is a good choice of F? We begin our search by demanding some simple properties of F

1. **Positivity**: F should be positive for positive valued arguments,

2. **Symmetry**: F should be symmetric in its arguments, and

3. **Monotonicity**: F should be monotonic in each of its arguments.

Positivity and symmetry are simple consequences of the definition of h. Monotonicity is a reasonable demand to make of F as one would expect that as the interaction between two vertices increases or decreases the strength of the hyperedge would be indicative of that change. Within these constraints there are still very many choices for F. We will consider the family of functions F_p parameterized by the positive scalar $p > 0$,

$$F_p(x_1, x_2, \ldots, x_u) \;=\; \left(\lambda(u) \sum_{i=1}^{u} x_i^p\right)^{1/p}, \quad u = \binom{k}{2}$$

where λ is a scalar function of the arity of F_p. We can now write Equation 3.1 as

$$h(z) \;=\; \left(\lambda\left(\binom{k}{2}\right) \sum_{\substack{v_i, v_j \in z \\ v_i < v_j}} g^p(v_i, v_j) \right)^{1/p}$$

For brevity we will write $\lambda(k) = \lambda\left(\binom{k}{2}\right)$. Using this and taking the p^{th} power on both sides gives us

$$h^p(z) \;=\; \lambda(k) \sum_{\substack{v_i, v_j \in z \\ v_i < v_j}} g^p(v_i, v_j) \tag{3.2}$$

We note that the above equation states that the $L_p$norm of the clique weights is proportional to the hyperedge weight. It is also worth noting that as the value of p increases, the $L_p$norm is biased towards the largest clique weight. For a given h and a fixed p this is a linear system in $g(v_i, v_j)^p$. Thus without any loss of generality we can restrict our analysis to the case $p = 1$. With this in mind let us interpret the above equation. Modulo a constant the above equation states that the weight of a hyperedge is the arithmetic mean of the edge weights in the clique it induces. Thus a natural choice for $\lambda(\binom{k}{2})$ is $\binom{k}{2}^{-1}$. Other choices for $\lambda(k)$ are possible and will amount to different weighting schemes when working with hyperedges of varying sizes. When working with hyperedges of the same size, which is the case in the current study the choice of $\lambda(k)$ is immaterial as it amounts to a uniform scaling of the graph weights. As the $Ncut$ algorithm is insensitive to scaling of the edge affinities this is not a problem. For the sake of concreteness we will use the arithmetic mean interpretation of the above equation and the resulting choice of $\lambda(k)$.

Also without loss of generality we will assume that the set of hyperedges has been ordered lexicographically based on the vertices incident on each hyperedge. A similar ordering is done on the set of graph edges too. We can now define the incidence matrix $\mathbf{\Delta}$. $\mathbf{\Delta}$ is a zero-one matrix, that represents the incidence relationship between a hyperedge in H and an edge in G. We say an

edge is incident on a hyperedge if the hyperedge contains both of its vertices.

$$\mathbf{\Delta}_{ij} \;=\; \begin{cases} 1 & \text{if edge } j \text{ is incident on hyperedge } i \\ 0 & \text{otherwise} \end{cases}$$

The rectangular matrix $\mathbf{\Delta}$ has $\binom{n}{k}$ rows and $\binom{n}{2}$ columns. Note that $\mathbf{\Delta}$ is an extremely sparse matrix with only $\binom{k}{2}$ non-zero entries. Now recall that d_2 denotes the vector of graph edge weights of size $\binom{n}{2}$, and d_k denote the vector of hyperedge weights. Then Equation 3.2 for the case of $p = 1$ can be written in matrix form as

$$d_k \;=\; \lambda(k)\mathbf{\Delta}d_2 \tag{3.3}$$

This equation assumes that $d_2 \geq 0$, hence when solving for d_2 given d_k, we will explicitly enforce this constraint. When working with hypergraphs with edge weights that are bounded above as in the case of affinities; we will enforce an upper bound $d_2 \leq 1$ also. Since the linear system is over-determined, the solution to Equation 3.3 has to be determined by minimizing the least squares error. Thus for the case of a hypergraph with hyperedge weights bounded in the interval $[0, 1]$, its graph approximation is given by the edge weight vector d_2 that minimizes the following constrained minimization problem

$$\min_{d_2} \|\lambda(k)\mathbf{\Delta}d_2 - d_k\|_F^2 \;, \quad 0 \leq d_2 \leq 1$$

The above optimization problem is an instance of the Bounds Constrained Least Squares problem. However as we noted earlier $\mathbf{\Delta}$ is a sparse matrix, and thus we can exploit efficient iterative methods for solving it [13]. We use `lsqlin` in MATLAB's Optimization Toolbox.

3.1.2 Partitioning the Hypergraph

We are now in a position to describe our proposed hypergraph partitioning algorithm. Given a dataset $\mathcal{D}$, the first step is the construction of the affinity hypergraph H by calculating the affinity

1. **H = generate_hypergraph(data)**
   Sample k-tuples from the dataset and compute the hyperedge weights for the corresponding hyperedges.

2. **G = clique_average(H)**
   Use Clique Averaging to construct the graph G that approximates H.

3. **clusters = normalized_cut(G)**
   Use the Normalized Cut algorithm to partition the vertices of G.

Figure 3.3: Hypergraph partitioning algorithm using Clique Averaging

for every distinct k-tuple in the dataset. However, calculating $\binom{n}{k}$ hyperedge weights can be computationally prohibitive. There are two ways to alleviate this problem: choosing a lower hyperedge order k, and subsampling less hyperedges than $\binom{n}{k}$. In many cases the user has a choice of the order of the hyperedge when constructing the hypergraph. For example, when clustering points into lines, it has been argued that at lease 3 points are needed, but a colinearity criteria could be established using more than 3 points. Using a simple counting argument one can show that since the number of within-cluster hyperedges to the number of between-cluster hyperedges goes down geometrically with increasing hyperedge size, the smallest possible value of k should be chosen. The second way to avoid the heavy computational cost is by subsampling the hypergraph and instead considering a hypergraph H' obtained by sparsely sampling hyperedges. Since the column rank of $\mathbf{\Delta}$ is $\binom{n}{2}$, we need at least $\binom{n}{2}$ rows, which in turn puts a lower bound on the number of hyperedges in H'. In our experiments we fix the number of subsampled hyperedges as $k^2\binom{n}{2}$ where k is the order of the hyperedge. We then use Clique Averaging to construct a graph G. To partition the graph into p parts, we use a spectral clustering algorithm that uses the first p eigenvectors of the normalized Laplacian of the graph and performs k-means clustering on the resulting k-dimensional embedding [41, 61, 74]. The entire algorithm is presented in Figure 3.3.

3.1.3 Duality

We now analyze the link between Clique Averaging and Clique Expansion. In Section 3.1.1 we saw that the graph edge weights as a result of Clique Expansion are given by

$$g(v_i, v_j) \;=\; \frac{1}{\mu(n,k)} \left(\sum_{v_i, v_j \in z} h(z) \right).$$

In light of the notation of the previous section we can re-write this as

$$\tilde{d_2} \;=\; \frac{1}{\mu(n,k)} \mathbf{\Delta}^\top d_k. \tag{3.4}$$

We use $\tilde{d}$ to indicate the graph edge weights from Clique Expansion. From Equation 3.3, the linear system for Clique Averaging is

$$\lambda(k) \mathbf{\Delta} d_2 \;=\; d_k \tag{3.5}$$

It is readily shown that the above two equations are duals of each other.

Let us now multiply both sides of Equation 3.4 by $\mathbf{\Delta}$ to get

$$\mathbf{\Delta} \tilde{d_2} \;=\; \frac{1}{\mu(n,k)} \mathbf{\Delta}\mathbf{\Delta}^\top d_k \tag{3.6}$$

Note that modulo a constant equations 3.5 and 3.6 differ only in the right hand side by a pre-multiplication by the matrix $\boldsymbol{S} = \mathbf{\Delta}\mathbf{\Delta}^\top$. To understand the action of this pre-multiplication let us consider the structure of the matrix $\mathbf{S}$.

Note that $\mathbf{S}$ is a symmetric matrix, with rows and columns corresponding to the hyperedges H. The entry in the z_i row and z_j column corresponds to the inner product of the z_i^{th} and the z_j^{th} rows of $\mathbf{\Delta}$. $\mathbf{\Delta}$ as we noted earlier is a zero-one matrix, hence the dot product counts the number of edges in the graph G that the two hyperedges share. These entries are easily calculated, for if $l = |z_i \cap z_j|$ denotes the number of vertices the two hyperedges have in common then $[\mathbf{S}]_{z_i z_j} = \binom{|z_i \cap z_j|}{2} = \binom{l}{2}$. Let the distance between two hyperedges of size k be $k - l$, then multiplication with the z_i^{th} row of

$\boldsymbol{S}$ is equivalent to multiplying each element of d_k by a decreasing function of the distance from the hyperedge z_i and summing over them. This is in fact a convolution of the hyperedge weights by a quadratically decreasing kernel. Thus $\mathbf{S}d_k$ is a low passed version of d_k. This implies that Clique Expansion solves the same approximation problem as Clique Averaging, but instead of operating on the original hypergraph, it operates on a low passed version of it. We know from basic signal processing theory that low pass filtering is an operation that loses information (only if low pass filtering actually cuts off high frequency information), and in the limit transforms the weight vector d_k into a constant vector. Hence, we argue that the approximation produced by Clique Averaging is of a higher quality and preserves the cluster structure present in the hypergraph H better.

3.1.4 Gibson's Algorithm

Gibson et al. have proposed a hypergraph partitioning algorithm which is designed for operating on categorical data in massive data bases [32]. While their original algorithm does not operate on weighted hypergraphs, a simple modification is needed for it to do so. Their algorithm is iterative in nature and the fixed points are the solution returned by the algorithm. They call these converged points as 'basins'. Let s_{ij} denote the value associated with the i^{th} vertex in the j^{th} basin. As before $h(z)$ denotes the weight the hyperedge z. We will use superscripts $^{(n)}$ to denote the iteration number. The iterative update can now be described in two steps as

1. $s_{ij}^{(n+1)} \;=\; \sum_{z\,;\,i\in z} \bigoplus_{k\in z, k\neq i} h(z) s_{kj}^{(n)}$
2. Perform Gram-Schmidt orthonormalization on the columns of the matrix $\left(s_{ij}^{(n+1)}\right)$

Gibson et al. discuss two choices for $\bigoplus$:

1. the product model: $\bigoplus_i x_i = \prod_i x_i$, and
2. the generalized sum model: $\bigoplus_i x_i = (\sum_i x_i^p)^{1/p}$.

The choice of the parameter p in Gibson et al's algorithm is very similar to the choice of p in our Clique Averaging. Basically Gibson's algorithm with the generalized sum model generates

the basins which are exactly the eigenvectors of the Laplacian of the graph that is generated by Clique Expansion algorithm. In the next section, its empirical performance is compared to other algorithms for both the product and generalized sum models.

3.2 Experiments

In this section we study the performance of six different algorithms out of which five are hypergraph partitioning algorithms. The sixth algorithm is a multi-round variant of RANSAC. We report the performance of the algorithms on two datasets. The algorithm are

1. Clique Averaging + Ncut (CAVERAGE)

   The hypergraph is approximated using Clique Averaging, and the resulting graph is partitioned using the normalized cut algorithm.

2. Clique Expansion + Ncut (CEXPAND)

   The hypergraph is approximated using Clique Expansion, and the resulting graph is partitioned using the normalized cut algorithm.

3. Gibson's Algorithm-Sum Model (GIBSONS)

   Gibson et al's algorithm operating under the sum model.

4. Gibson's Algorithm-Product Model (GIBSONP)

   Gibson et al's algorithm operating under the product model.

5. kHMeTiS (KHMETIS)

   The leading tool for hypergraph partitioning in the VLSI community based on multi-level iterative refinement.[2]

6. Cascading RANSAC (CRANSAC)

   A simple multi-round extension to the RANSAC algorithm. In the i-th round a number of

[2]The implementation of kHMeTis algorithm can be downloaded as a part of the hMETIS package; http://www-users.cs.umn.edu/~ karypis/metis/hmetis/index.html .

trials are performed to identify that k-tuple that has the highest number of inliers. This k-tuple and its associated inliers are identified as the i-th group in the dataset and removed from it.

Reporting unbiased performance comparison of clustering algorithms is a hard problem, since each algorithm that one compares against has one or more free parameters that need to set according to the particular problem at hand. Thus while comparing performance across problems, an approach giving each algorithm the best shot would need to perform a sweep over all possible parameter values. While this might report the best behavior for an algorithm it is clearly not informative about the robustness of the algorithm to parameter choice, a property that is of vital importance to a user who is using the algorithm on a novel dataset. Thus it is important to use an experimental protocol that is as close as possible to real world usage.

One of the ways in which algorithms are tuned is by running them on a small pilot dataset similar to the real problem. This is the basis of our experimental protocol. When running an algorithm over a suite of experiments, we choose a problem that lies at the center of the set of experiments in terms of complexity and choose the best performing parameters using a parameter sweep. This parameter setting is used for all the experiments in the test suite. To be fair to CRANSAC in terms of computation resources, we set the total number of trials it could perform to be equal to the number of hyperedges in the corresponding hypergraph. GIBSONS and CAVERAGE were run with $p = 1$. The only free parameter across all the hypergraph partitioning algorithms was the parameter σ that was used to convert a dissimilarity d into the affinity $a = e^{-d/\sigma}$. In the case of CRANSAC the error threshold for inlier detection was the free parameter.

3.2.1 k-lines Clustering

In the first experiment, we consider the k-lines problem in spaces of dimension greater than two, i.e., given a set of points in $\mathbb{R}^d$, the task then is to partition them into a number of d-dimensional lines. In the case of lines in two dimensions the Hough transform solves this problem quite effec-

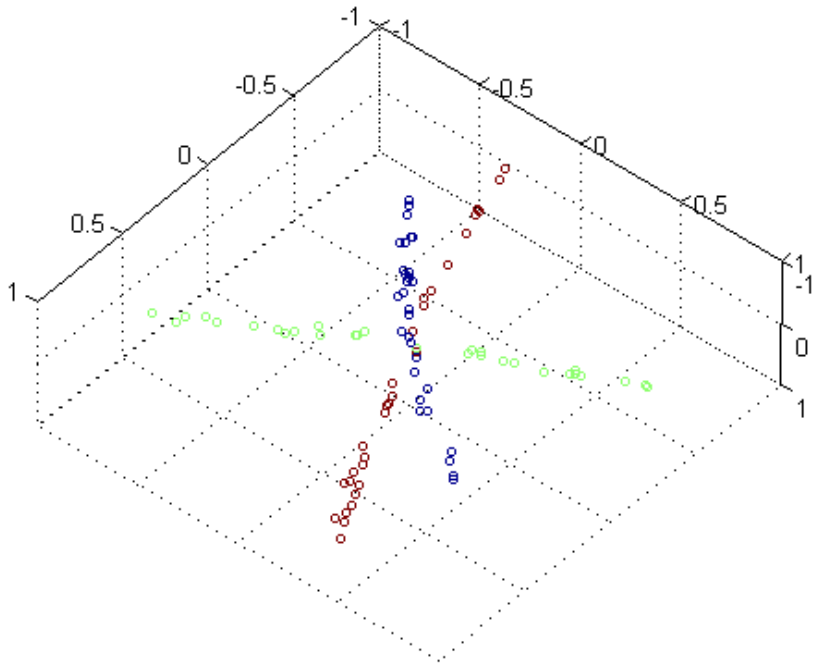


Figure 3.4: k-lines problem

tively, but with three or more dimensions there is no convenient parameterization that can be used. Pairwise measures of similarity are not applicable here since any two points are co-linear, thus it takes at least three points to determine a measure of co-linearity. This is an example of a triadic relationship. The dissimilarity measure on triples of points is their distance to the best fitting line. Our dataset consists of points sampled from gentle curves with noise added to them. All the curves pass through the origin. Thus all clusters overlap with each other to some degree. The curves are generated as arcs of circles with a controllable radius of curvature (see Figure 3.4). We used curves instead of lines to see how the algorithms behave when the model is not exact but only an approximation of the true model. Let us consider the performance of the six algorithms on a dataset containing 5 curves, in the cube $[-1, 1]^5$ in five dimensions. We sample 70 points from each curve for a total of 350 points. The hypergraph was generated by sampling $k^2\binom{n}{2} = 549675$ 3-tuples. For this dataset we considered the performance of the six hypergraph partitioning algorithms over varying values of σ. The average clustering error of each algorithm is shown in Table 3.1.

CAVERAGE	CEXPAND	GIBSONS	GIBSONP	KHMETIS	CRANSAC
12.6%	12.9%	17.3%	55.1%	18.0%	23.4%

Table 3.1: Clustering results on the k-lines database using various hypergraph partitioning methods. The average clustering error of each algorithm is shown.

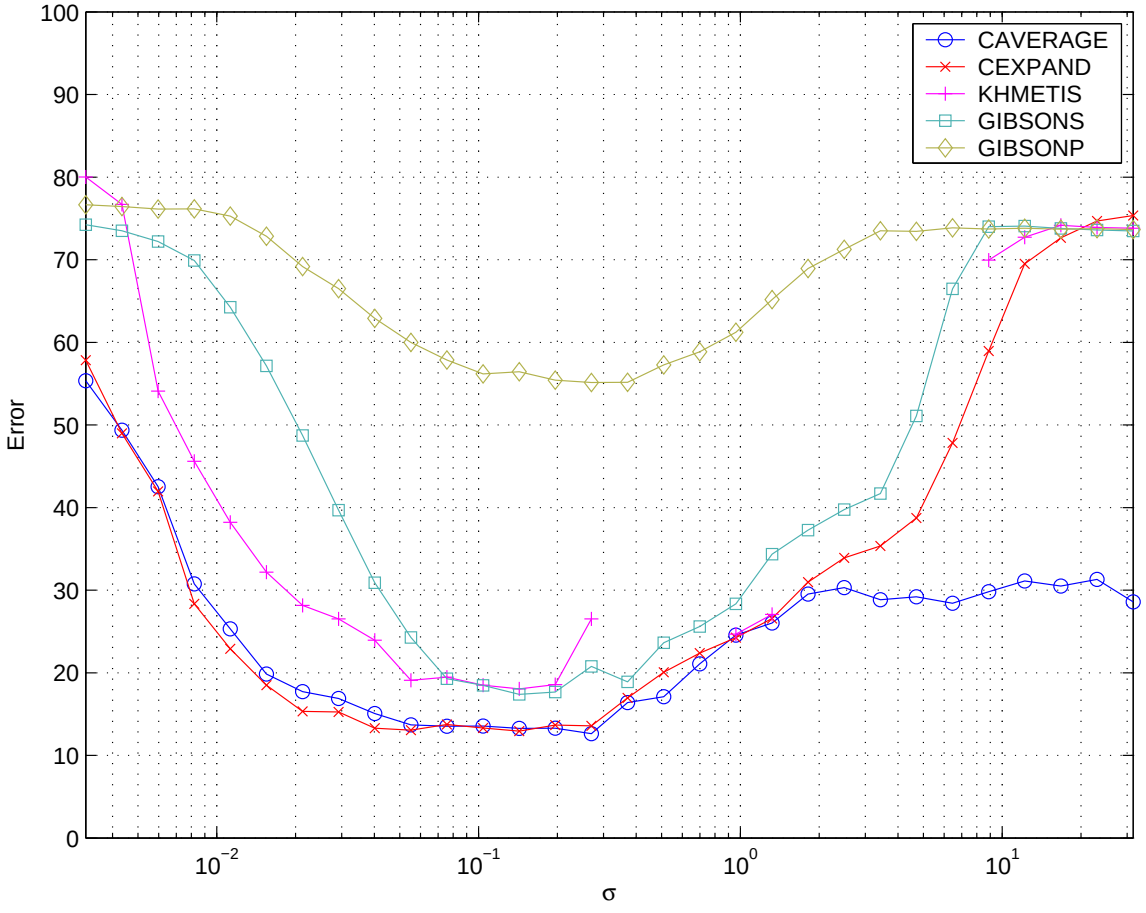


Figure 3.5: k-lines parameter sweep

But a more elaborate picture emerges when one looks at the performance of the algorithms over a range of values of σ. Figure 3.5 plots this behavior. The graph has a number of notable features. We begin by noting that Clique Expansion and Clique Averaging are the two best performing algorithms, and for small and moderate values of sigma there is virtually no difference between their performance. It is however interesting to note that as sigma increases further the performance of Clique Expansion sharply degrades and reaches 80% error which is the same as chance. Clique Averaging on the other hand continues to perform well at about 30% error while the other four algorithms are operating at $70\% - 80\%$ error. The error curve for HMETIS is disjoint because for missing values of σ the program crashed.

3.2.2 Clustering over Illumination Variation

It has been shown that all the images of a convex Lambertian object illuminated by a point light source without shadowing must lie in a three dimensional subspace [72]. Belhumeur and Kriegman later introduced the concept of the 'illumination cone', which yields a very effective image clustering algorithm (in Chapter 4), but the subspace constraint is used in this experiment to show the effectiveness of the proposed algorithm.

The three-dimensional subspace constraint leads to a natural definition of a measure of dissimilarity over four or more images and allows us to perform clustering using it. Indeed this is a generalization of the k-lines problem to the k-subspaces problem. If we assume that the four images under consideration form the columns of a matrix, then $d = \frac{s_4^2}{s_1^2+\cdots+s_4^2}$ serves as a measure of dissimilarity. Here s_i is the i-th singular value of this matrix. Images of ten individuals in the Yale B face database [31] (10×45 images in total, see Figure 3.6) are chosen for the experiment. The aim of the clustering procedure is to partition the images into groups such that no group contains images of two different people.

Figure 3.7 shows the result of performing a parameter sweep over the parameter σ for a set of images of seven people. The gross behavior of the algorithms in Figure 3.5 and Figure 3.7 is very similar. Again CAVERAGE and CEXPAND are consistently the best performing algorithms, and

(A)

(B)

Figure 3.6: Face dataset (A) 10 people from Yale B dataset (B) images of one person under 45 different lighting conditions.

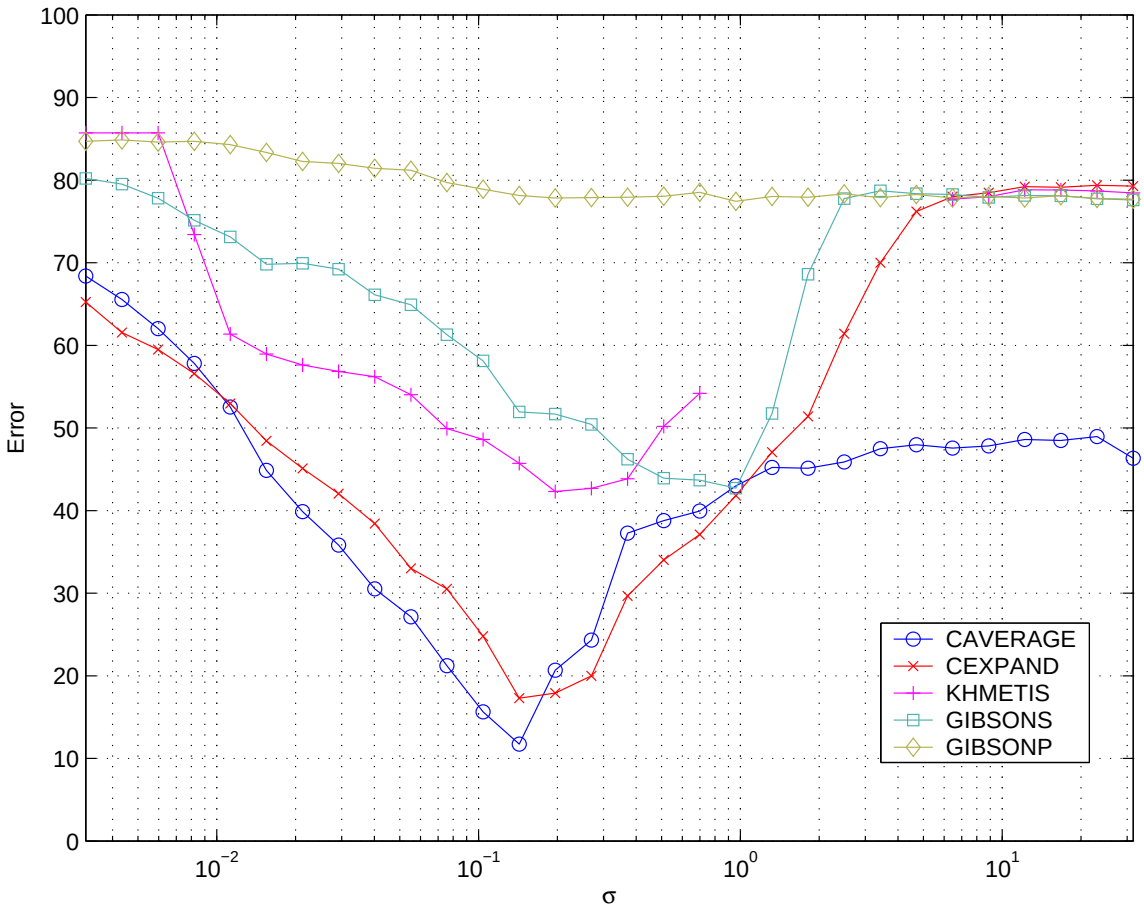


Figure 3.7: Yale database parameter sweep

	4 people	6 people	8 people	10 people
AVERAGE	4.2% (6.3)	12.7% (8.4)	17.4% (4.0)	16.0% (3.0)
EXPAND	11.8% (3.4)	17.6% (5.4)	21.8% (5.4)	24.9% (4.3)
GIBSONS	25.9% (7.3)	42.2% (3.8)	47.7% (3.0)	51.5% (2.1)
GIBSONP	67.4% (2.3)	75.2% (1.2)	79.7% (0.8)	82.8% (0.7)
KHMETIS	21.5% (4.3)	41.9% (6.8)	38.4% (4.7)	58.3% (3.3)
CRANSAC	16.2% (9.5)	23.6% (9.2)	35.1% (7.9)	37.1% (6.6)

Table 3.2: Clustering results on Yale B face database using various hypergraph partitioning methods. The average and standard deviation of clustering errors from 30 trials are shown.

CAVERAGE is much more robust to changes in the value of the scaling parameter σ. This experiment was used as the basis for tuning the parameters for individual algorithms for the following experiment.

In the following table we present the results of running the six algorithms on four subsets of the Yale face dataset with increasing number of images and people. Each algorithm was run 30 times with the parameter. The results are in the form 'mean error (standard deviation)'. As can be seen in Table 3.2, Clique Averaging beats all other algorithms across the board.

While two problem sets do not make for conclusive evidence, they are indicative of a few general trends. Clique Averaging is much less sensitive to changes in the dynamic range of hyperedge weights, providing empirical verification of the relationship established between Clique Expansion and Clique Averaging in Section 3.1.3. It is also consistently the best performing algorithm amongst the six we have tested.

3.3 Discussion

In this work we have introduced hypergraph partitioning as the natural formulation for a number of computer vision tasks. A new hypergraph approximation algorithm is introduced and its better performance than other existing algorithms is shown both by theoretical argument and by experiments. We also compared the performance of our proposed algorithm to four existing hypergraph partitioning algorithms and a multi-round variant of RANSAC. In all experiments, Clique Averag-

ing algorithm outperformed its competitors both in terms of clustering error as well as insensitivity to parameter changes.

There do however remain a number of open questions and directions for future work. The most important question is that of computational complexity. Since we solve for all the graph edge weights, the sampling complexity for the algorithm is lower bounded by $O(n^2)$. However there is evidence that for data that is clusterable into a small number of clusters, spectral clustering can be performed using far fewer than $O(n^2)$ graph edges [27], thus it seems that a significant reduction in the sampling complexity of Clique Averaging is possible.

While the methods we discussed in this chapter are focused on converting a hypergraph to a graph and then operating upon it, the development of methods that operate directly on the hypergraph without any intermediate or implicit reduction to a graph remains an open question. One would expect that such direct methods would perform better and be more robust than their approximation-based counterparts.

Govindu introduced another hypergraph partitioning algorithm in tensor theoretic approach [35]. His algorithm computes a decomposition a tensor which contains hyperedge weights using the higher-order SVD algorithm. Instead of working with a full tensor, he proposed a randomized algorithm that randomly samples columns of the flattened matrix of the tensor, and builds a probability matrix which can be used as an affinity matrix. This algorithm is very different from Clique Averaging in many ways and further study is needed in comparing and possibly unifying two algorithms.

Chapter 4

Image Clustering Over Varying Illumination

In this chapter, we introduce two appearance-based methods for clustering a set of images of 3D objects acquired under varying illumination. This problem can be formulated as, given a set of unlabeled images of a collection of objects acquired under different lighting conditions, decompose the set into disjoint subsets corresponding to individual objects. In Chapter 3 one solution for this problem using a hypergraph approximation was presented, and in this chapter, more problem specific issues (about images under different lighting conditions) are discussed in order to develop more effective image clustering algorithms.

For a Lambertian object, it has been proven that the set of all images taken under all lighting condition forms a convex polyhedral cone in the image space [10], and this polyhedral cone can be approximated well by a low-dimensional linear subspace [8, 23, 65]. Recall that a polyhedral cone in $\mathbb{R}^N$ is defined by a finite set of generators (or extreme rays) $\{x_1, \cdots, x_d\}$ such that any point x in the cone can be written as a linear combination of $\{x_1, \cdots, x_d\}$ with *non-negative* coefficients. With these observations, the k-class clustering problem for a collection of images $\{I_1, \cdots, I_n\}$ can be cast as finding k polyhedral cones that best fit the data. Ultimately what we need to do is to assign a non-negative number a_{ij}, called the conic affinity, for each pair of images I_i and I_j. Intuitively, a_{ij} measures how likely I_i and I_j come from the same polyhedral cone, which refers to the same object.

While the algorithm outlined above exploits the hidden geometric structures (the cones) in the image space, the second algorithm exploits the effect of the 3D surface geometry of a Lambertian object on its appearances under varying illumination. Chen et al. [17] have shown that there is no such notion of illumination invariants that can be extracted from an image. However, they also demonstrate that image gradients can be utilized in a probabilistic framework to determine the likelihood of two images originating from the same object. The important conclusion of their work is that while the lighting direction can be arbitrary, the direction of the image gradient is not. The second algorithm directly utilizes this illumination insensitive property of the image gradient vector. For a pair of images, we define the gradient affinity as follows. The image gradients at each pixel in both images are first computed. The magnitude and orientation of the image gradient vectors at the corresponding pixels are compared, and the results are aggregated over the entire image to form the gradient affinity.

Unlike some clustering problems [47] studied earlier, the clustering problem studied in this work benefits greatly from many structural results concerning illumination effects that emerged in the past few years, e.g., [8, 10, 65]. The first algorithm computes the affinity measures globally in the sense that the affinity between any pair of images is actually determined by the entire collection, since the algorithm operates directly on the underlying geometric structures in the image space, i.e., the illumination cones. This global characteristic is the major difference from other affinity measures commonly defined in other clustering and segmentation problems, e.g., [47, 74]. The second algorithm, more akin to the usual approach in the literature, computes the affinity between a pair of images using just two images. Both methods are straightforward to implement. Also note that both algorithms differ from other image clustering work [29, 34, 57, 70, 83] in that they treat each image as a whole thing instead of trying to find some local features in it.

It is clear from Figure 4.5 and Figure 1.3 that a direct approach using the usual L^2-distance metric coupled with standard clustering techniques will not yield promising results. However, this work shows that it is precisely the use of these subtle structural or probabilistic cues which is the gist of the problem; simple and effective solutions can be designed by appealing directly to

these structural results. We will demonstrate experimentally that these two simple algorithms are surprisingly effective when applied to clustering large collections of unlabeled images.

4.1 Similarity Measures

In this section, we detail the two proposed clustering algorithms. Schematically, they are similar in their overall algorithmic structure to other pairwise clustering algorithms, e.g., [9, 74]. That is, we define similarity measures between all pairs of images. These similarity or affinity measures are represented in a symmetric matrix $A \in \mathbb{R}^{n \times n}$, i.e., the affinity matrix. The second step is a straightforward application of any standard spectral clustering method [61, 84]. The novelty of our clustering algorithms lie in the definition of the two affinity measures described below. The conic affinity and the gradient affinity will be introduced in the first two subsections, and they will be compared with previous work in the last subsection.

Let $\{I_1, \cdots, I_n\}$ be a collection of unlabeled images. In the subsequent discussion, n and k will always denote the number of sample images and the number of clusters, respectively. We assume:

1. The images were taken from k different objects with *Lambertian* reflectance. That is, there is an assignment function $\rho : \{I_1, \cdots, I_n\} \rightarrow \{1, \cdots, k\}$.

2. For each cluster of images, $\{\, I_i \mid \rho(I_i) = j, 1 \leq j \leq k \,\}$, all images were taken from the same viewpoint (i.e., relative position and orientation between the object and the camera). However, the external illumination conditions under which the images were taken may vary widely.

3. All images have the same number of pixels, N.

4.1.1 Conic Affinity

Let $\mathcal{D} = \{x_1, \cdots, x_n\}$ be points in the image space $\mathbb{R}^N$ obtained by raster scanning the images. As mentioned in the introduction, the clustering problem is equivalent to determining a set of k polyhedral cones that best fit the input data. However, it is almost impossible to search for such a set of k polyhedral cones directly in the high-dimensional image space.

The first step of our algorithm is to define a good metric of the likelihood that a pair of points come from the same cone. In other words, we want a numerical measure that can detect the underlying conic structure in the high-dimensional image space. Recall that at a fixed pose, the set of images of a convex Lambertian object under all possible illumination conditions forms a polyhedral cone, and any image in the cone can be written as a non-negative linear combination of the cone's generators. For each point x_i, we seek a non-negative linear combination of all the other input samples that approximates x_i. In other words, we find non-negative coefficients $\{b_{i1}, \cdots, b_{i(i-1)}, b_{i(i+1)}, \cdots, b_{in}\}$ such that

$$x_i \;=\; \sum_{j \neq i}^{n} b_{ij}\, x_j \;, \quad b_{ij} \geq 0,\; b_{ii} = 0 \tag{4.1}$$

in the least square sense.

Let $\{y_1, \cdots, y_d\}$ be a subset of the collection $\mathcal{D}$. If x_i actually belongs to the cone generated by this subset, this will imply that $b_{ij} = 0$ for any x_j not in the subset. If x_i does not belong to the cone yet lies close to it, x_i can be decomposed as the sum of two vectors $x_i = \bar{x}_i + r_i$ with $\bar{x}_i$, the projection of x_i on the cone, and r_i, the residue of the projection. Clearly, $\bar{x}_i$ can be written as a linear combination of $\{y_1, \cdots, y_d\}$ with non-negative coefficients. For r_i, because of the *non-negative* constraint, the non-negative coefficients in the expansion

$$r_i \;=\; \sum_{j \neq i}^{n} b^r_{ij} x_j.$$

will be dominated by the magnitude of r_i. This follows from the following simple proposition.

Note that the proposition is false without non-negative constraint on the coefficients.

Proposition 1. *Let I and $\{I_1, \cdots, I_d\}$ be a collection of images. If I can be written as a linear combinations of $\{I_1, \cdots, I_d\}$ with non-negative coefficients:*

$$I \;=\; \alpha_1 I_1 + \cdots + \alpha_d I_d$$

where $\alpha_i \geq 0$ for $1 \leq i \leq d$, then $\alpha_i \leq (I \cdot I_i)/\|I_i\|^2$ and $\alpha_i \leq \|I\|/\|I_i\|$.

Proof. Since $I \;=\; \alpha_1 I_1 + \cdots + \alpha_d I_d \;\;\geq\;\; \alpha_i I_i$,

$$\alpha_i \;\leq\; \frac{I \cdot I_i}{\|I_i\|^2} \;\leq\; \frac{\|I\|\|I_i\|}{\|I_i\|^2} \;=\; \frac{\|I\|}{\|I_i\|}$$

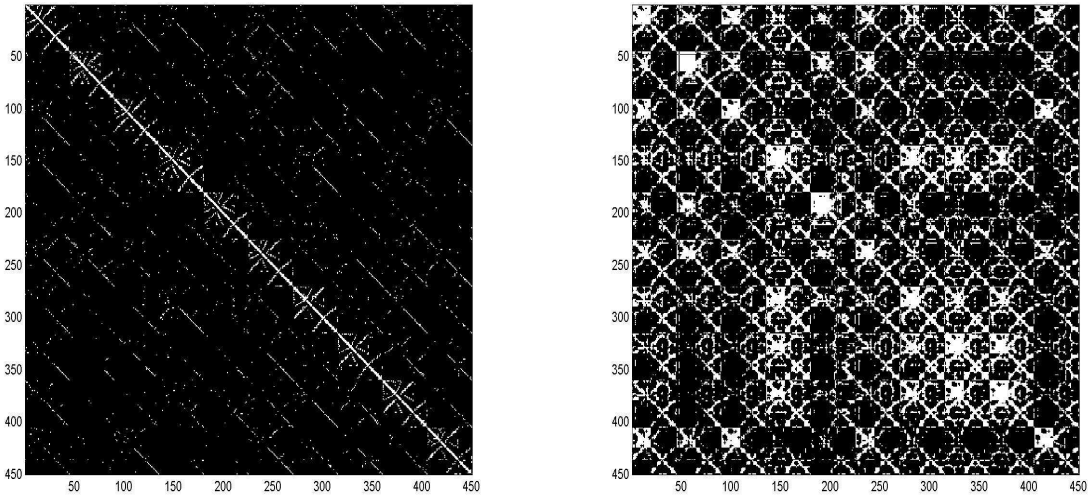
□

Therefore, we expect the coefficients in the expansion of x_i to reflect the fact that if x_i were well-approximated by a cone generated by $\{y_1, \cdots, y_k\}$, then the corresponding coefficients b_{ij} would be large (relatively) while others would be small or zero. That is, the coefficients in the expansion should serve as good indicators of the hidden conic structures.

Another important characteristic of the non-negative combinations is that there are only a few coefficients that are significant in magnitude. Typically there are only a few nonzero b_{ij} in Equation 4.1. This is indeed what has been observed in our experiments as well as prior work on non-negative matrix factorization [55]. Figure 4.1 shows coefficients of the affinity matrix A (defined below) computed with and without non-negative constraints using a set of 450 images from the Yale B database.

We form a matrix B by taking the coefficients in the expansion in Equation 4.1 as the entries of $B = (b_{ij})$. We normalize each column of B so that the sum is 1. This step ensures that the overall contribution of each input image is the same. By construction, $b_{ij} \neq b_{ji}$ in general, i.e., the B matrix is not symmetric. So we symmetrize B to obtain the affinity matrix $A = (B + B^\top)/2$.

The computational complexity of the algorithm is dominated by the computation of the non-negative least square approximation for each point in the collection. For a collection with a



(A) (B)

Figure 4.1: Affinity matrices A using (A) non-negative linear least square approximations, and (B) the usual linear least square approximation without non-negativity constraints. Brighter color refers larger similarity. Compared to (B), large similarities form (weak) block diagonal structure in (A).

large number of points, solving the least square approximation for every single element is time-consuming. Therefore, we introduce a parameter m which gives the maximum number of images used in non-negative linear least squares estimation. That is, we only consider the m closest neighbors of x_i in computing Equation 4.1. Here, the distance involved in defining neighbors can be taken to be any similarity measure. The L^2-distance metric is sufficient for the clustering task considered in this work.

The proposed algorithm, summarized in Figure 4.2, is very easy to implement, and the clustering portion of the algorithm takes less than twenty lines of code in Matlab. The last step involves an optional k-subspace clustering algorithm which we will discuss in Section 4.2.1.

4.1.2 Gradient Affinity

In the previous subsection, we have explored the possibilities of defining affinity by exploiting the hidden conic structures in the image space. In this section, we explore the possibilities of defining

1. **Non-negative Least Square Approximation**
   Let $\{x_1, \cdots, x_n\}$ be the collection of input samples. For each input sample x_i, compute a non-negative linear least square approximation of x_i by all the samples in the collection except x_i

$$x_i \approx \sum_{j \neq i} b_{ij} x_j$$

   with $b_{ij} \geq 0, \forall j \neq i$ and set $b_{ii} = 0$. Normalize $\{b_{i1}, \cdots, b_{ik}\}$:

$$b_{ij} = \frac{b_{ij}}{\sum_l b_{il}}.$$

   (If N is too large, use only m closest neighbors of x_i for the approximation.)

2. **Compute Affinity Matrix**

   (a) Form the matrix $B = (b_{ij})$.

   (b) Define $A = (B + B^\top)/2$.

3. **Spectral Clustering**
   Using A as the affinity matrix, apply any standard spectral method.

4. **(Optional) k-subspace Clustering**
   Apply k-subspace clustering to further exploit the linear geometric structures hidden among the images.

Figure 4.2: Clustering algorithm based on conic affinity.

affinities using the object's 3D geometry. The effect of the object's geometry on its images taken under different illumination conditions has been analyzed in great detail in Chen et al. [17]. There, it has been shown that the important quantity to compute for studying illumination variation is the image gradient. For a Lambertian surface, the image gradient ∇I depends on the object geometry (surface normal $\mathbf{n}$) and the albedo α as:

$$\nabla I = (\hat{u}\,\kappa_u\,\mathbf{s}_u + \hat{v}\,\kappa_v\,\mathbf{s}_v) + (\nabla\alpha)\,\mathbf{s}\cdot\mathbf{n}$$

Here, the $\hat{u}$ and $\hat{v}$ are local tangential directions defined by the principal directions, κ_u and κ_u are the two principal curvatures, and $\mathbf{s}$ is the lighting direction. Further analysis based on this

equation has shown that the magnitudes and orientations of the image gradient vectors form a joint distribution which can be utilized to give the likelihood of two images originating from the same object.

We take a simpler approach using direct comparison between image gradients. That is, we sum over the image plane the differences in the magnitude of the image gradient and the relative orientation (i.e., angular difference between the two corresponding image gradients). Given a pair of images I_i and I_j, let ∇I_i and ∇I_j denote their image gradients, and $\nabla I(w)$ be the image gradient vector at pixel w. First, we define M_{ij} as the sum over all pixels of the squared-differences between the magnitudes of ∇I_i and ∇I_j.

$$M_{ij} \;=\; \sum_{w=1}^{N} \left(\|\nabla I_i(w)\| - \|\nabla I_j(w)\|\right)^2$$

Next, we calculate the difference in orientation. O_{ij} is defined as the sum over all pixels of the squared-angular differences

$$O_{ij} \;=\; \sum_{w=1}^{N} \left(\angle(\nabla I_i(w), \nabla I_j(w))\right)^2$$

Prior to computing the gradients, the image intensities are normalized to $\{0, 1\}$, and the angular difference between the two image gradients are also normalized from the range of $\{-\pi, \pi\}$ to $\{0, 1\}$. The method is summarized in Figure 4.3. The algorithm is again very easy to implement.

4.1.3 Comparison with previous work

One previous clustering algorithm that shares some similarities with ours is the work by Basri et al. [9]. Both methods exploit the underlying geometric structures in the image space: appearance manifolds [60] vs. illumination cones [10]. However, the conic affinity approach differs fundamentally in one crucial aspect: their method is based on *local* geometry while ours is based on *global* characteristics. This is because the geometric structures for which the two algorithms operate are

1. **Compute Image Gradients**
   Let $\{I_1, \cdots, I_n\}$ be the collection of input images. Let ∇I_i denote the image gradient of I_i. For $1 \leq i, j \leq n$, define

$$\begin{aligned} M_{ij} &= \sum_{w=1}^{N} (\|\nabla I_i(w)\| - \|\nabla I_j(w)\|)^2 \\ O_{ij} &= \sum_{w=1}^{N} (\angle(\nabla I_i(w), \nabla I_j(w)))^2 \end{aligned}$$

   where N is the number of pixels in the images.

2. **Compute Affinity Matrix**
   Set the entries of the affinity matrix A as

$$A_{ij} = \exp -\frac{1}{\sigma}(M_{ij} + O_{ij})$$

   for some real number σ.

3. **Spectral Clustering**
   Using A as the affinity matrix and apply any standard spectral method for clustering.

4. **(Optional) k-subspace Clustering**
   Apply k-subspace clustering to further exploit the linear geometric structures hidden among the images.

Figure 4.3: Clustering algorithm based on gradient affinity

different.

The algorithm proposed by Basri et al. [9] deals mainly with clustering problems with pose variation but under fixed illumination conditions. The affinity is computed based on local linear subspaces represented by the tangent planes of the appearance manifold. The non-linear nature of the appearance manifold is reflected by the local affinity measures in the absence of a global linear structure. Note that under similar lighting conditions, the shadow formations on different faces are roughly the same. This implies that the tangent space estimation in Basri's algorithm would produce tangent planes with tangent vectors equal to zeros in the shadowed region. This is, put in their terminology, images taken under the same lighting conditions are more likely to have close to parallel tangent planes.

To cluster these images according to identity correctly, the algorithm needs to find polyhedral cones for individuals, and each image in a cone can be represented as a convex combination of basis images (generators of the cone). Given an image I, our algorithm considers all the other images in order to find the set of images (i.e., the ones in the same illumination cone) that best reconstruct I. However, this cannot be realized by the approach in Basri's algorithm which operates on a pairwise basis.

4.2 Clustering Algorithms

In Chapter 2, we briefly summarized the spectral methods [61]. The standard spectral clustering algorithm uses the k-means clustering algorithm to produce the final clustering result. Here we propose another clustering algorithm called k-subspace clustering algorithm, which uses subspace distance instead of the standard L^2 distance.

4.2.1 k-subspace Clustering

A typical spectral clustering method analyzes the eigenvectors of an affinity matrix of data points, where the last step often involves thresholding or grouping. For the clustering problem considered

in this work, we know that the data points come from a collection of convex cones which can be approximated well by low dimensional linear subspaces. Therefore, each cluster should also be well-approximated by some low-dimensional subspace. We therefore exploit this peculiar aspect of the problem and supplement with one more clustering step on top of the results obtained from spectral analysis. The algorithm we are using is a variant of the usual k-means clustering algorithm. While the k-means algorithm basically finds k cluster centers using a point-to-point distance metric, the task here is to find k linear subspaces using a point-to-plane distance metric.

The k-subspace clustering algorithm, summarized in Figure 4.4, iteratively assigns points to the nearest subspace (cluster assignment), and for a given cluster, it computes a subspace that minimizes the sum of the squares of distance to all points of that cluster (cluster update). Similar to the k-means algorithm, the k-subspace clustering method will terminate after a finite number of iterations. This is the consequence of the following two simple observations:

1. There are only finitely many ways that the input data points can be assigned to k clusters.

2. Define an objective function (of a cluster assignment) as the sum of the square of the distance between all points in a cluster and the cluster subspace. It is obvious that the objective function decreases during each iteration.

The result of the k-subspace clustering algorithm depends very much on the initial collection of k subspaces. Typically, as for k-means clustering also, the algorithm only converges to some local minimum which may be far from optimal. However, after applying the clustering algorithm using either the conic or gradient affinity, we have a new assignment function $\tilde{\rho}$, which is expected to be close to the true assignment function ρ. We will use $\tilde{\rho}$ to initiate the k-subspace algorithm by replacing the assignment function ρ in the Cluster Assignment step (see Figure 4.4) with $\tilde{\rho}$.

Note that the k-subspace problem can also be expressed in terms of the multi-adic clustering problem discussed in Chapter 3. The main difference in the approach proposed in this section is that it uses the model-based clustering instead of the graph-based. Usually model-based algorithms runs faster and returns plausible results given that the initialization is close to the desired goal.

1. **Initialization**
   Starting with a collection $\{S_1, \cdots, S_k\}$ of k subspaces of dimension d, where $S_i \in \mathbb{R}^s$. Each subspace S_i is represented by one of its orthonormal bases, U_i.

2. **Cluster Assignment**
   We define an operator $P_i = \mathrm{I}_{d\times d} - U_i$ for each subspace S_i. Each sample x_i is assigned a new label $\rho(x_i)$ such that
   $$\rho(x_i) \;=\; \arg\min_q \|P_q(x_i)\|$$

3. **Cluster Update**
   Let S_i be the scatter matrix of the sampled labeled as i. We take the eigenvectors correspond to the top d eigenvalues of S_i. The eigenvectors corresponding to the top d eigenvalues are the orthonormal basis, $U_i^{'}$ of the $S_i^{'}$. Stop when $S_i^{'} = S_i$. Otherwise, go to Step 2.

Figure 4.4: k-subspace Clustering Algorithm.

4.3 Experiments and Results

We performed numerous experiments using the Yale and the CMU databases, and compared them with other clustering algorithms. Also the experimental validation on the effectiveness and robustness of the proposed algorithms are presented, and the comparison between two proposed algorithm is briefly described.

4.3.1 Datasets

The Yale database B that we used for experiments comprises two subsets of face images in frontal and side pose [31]. Each subset consists of 450 face images where 45 frames of the same person are acquired under varying lighting. The only difference between these two subsets is the pose at which the images were acquired. Figure 4.5.A and 4.5.B shows sample images of two persons from these subsets. Each image is then manually cropped and downsampled to 21×24 pixels for computational efficiency. The CMU PIE database [75] used in our experiments consists of the subset where frontal view images of the subjects were taken under different illumination conditions

(A)

(B)

(C)

Figure 4.5: Sample images acquired at frontal view (A) and a non-frontal view (B) in the Yale database B, and frontal view (C) in the CMU PIE database.

Method	Yale B (frontal)	Yale B (pose)	PIE 66 (frontal)
Conic + spectral + k-subspace	0.44	4.22	4.18
Conic + spectral + k-means	0.89	6.67	4.04
Gradient + spectral + k-means	1.78	2.22	3.97
Linear + spectral + k-subspace	62.44	58.00	69.19
Spectral clustering	65.33	47.78	32.03
k-subspace	61.13	59.00	72.42
k-means	83.33	78.44	86.44

Table 4.1: Clustering results using various methods.

but without background light (PIE 66: each person has 21 images). (See Figure 4.5.C for sample images from PIE 66 subset). Note that this is a more difficult subset than the other subset in the CMU PIE data set where images were taken with ambient background light. Similar to pre-processing with the Yale dataset, each image is manually cropped and downsampled to 21×24 pixels. Clearly the large appearance variation of the same person in these data sets makes the face recognition problem rather difficult [31, 76], and thus the clustering problem extremely difficult. Nevertheless we will show that our methods achieve very good clustering results, and outperform other algorithms.

4.3.2 Results and Comparison with Other Clustering Algorithms

We tested several clustering algorithms with different sets of parameters, where we further assume that the number of clusters k is known. Recent results on spectral clustering algorithms show that it is feasible to select an appropriate k value by analyzing the eigenvalues or eigenvectors [18, 61, 63, 84, 88]. For the k-means and k-subspace algorithms (last two rows in Table 4.1), the parameters were empirically tuned and experiments were repeated to get average results since they are sensitive to initialization and parameter selections, especially in the high-dimensional space. Table 4.1 summarizes the experimental best results achieved by each method:

1. **Conic + spectral + k-subspace**:

   the proposed conic affinity method with spectral analysis and k-subspace clustering method

Method	Yale B (Pose) subject 1-5	Yale B (Pose) subject 6-10
Conic + spectral + k-subspace	0.0	0.0
Gradient + spectral + k-means	8.9	6.67

Table 4.2: Clustering results with high-resolution images.

2. **Conic + spectral + k-means** :

   the conic affinity method with spectral analysis and k-means clustering

3. **Gradient + spectral + k-means** :

   the proposed gradient affinity method with the standard spectral clustering

4. **Linear + spectral + k-subspace**:

   linear decomposition without the non-negative constraint, followed by spectral analysis and k-subspace method

5. **Spectral clustering**, **k-subspace** and **k-means** :

   straightforward application of spectral clustering, k-subspace, and k-means clustering.

The error rate is computed based on the number of images that are not clustered to the group of the same identity as we have the ground truth about each image in these data sets.

Our experimental results shown in Table 4.1 suggest a number of conclusions. First, the results clearly show that the methods using conic or gradient affinity outperform other methods by a large margin. Comparing the results on rows 1 and 4, they show that the non-negative constraints play an important role in achieving good clustering results. Second, the proposed conic and gradient affinity metric facilitates spectral clustering method in achieving very good results. The use of k-subspace further improves the clustering results after applying conic affinity with spectral methods. (See also Figure 4.6). Finally, a straightforward application of the k-subspace algorithm does not work well in the high-dimensional image space.

To further analyze the strength of the conic and gradient affinities, we apply these metrics to high-resolution images of 168×184 pixels (i.e, image size before downsampling in previous exper-

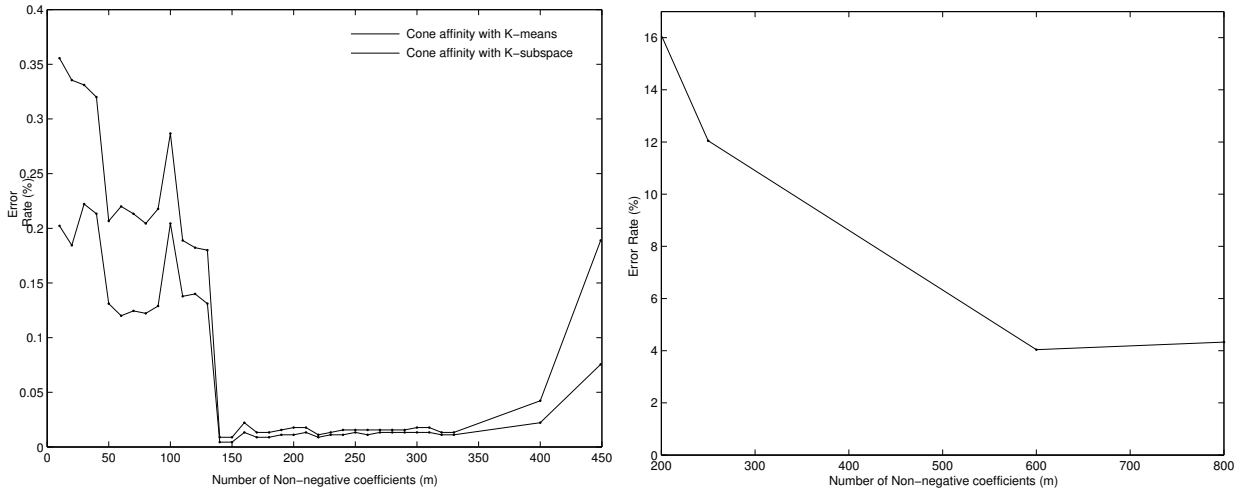
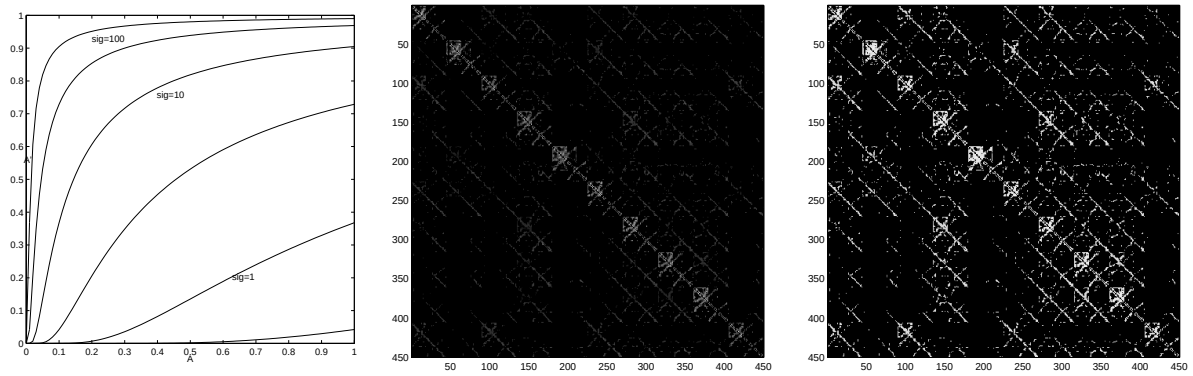


Figure 4.6: Effects of parameter selection on clustering results with the Yale database B and PIE66 (frontal) database.

iments). Table 4.2 shows the experimental results using the non-frontal images of the Yale database B. For computational efficiency, we further divide the Yale database B into two sets. The results demonstrate that the conic affinity metric with spectral clustering renders perfect results. The experiments also show that applying the gradient affinity metric to low-resolution images render better clustering results than the results in high-resolution images. This suggests that computation of gradient metric is more reliable in low-resolution images and surprisingly such information is sufficient for the clustering task considered in this work.

4.3.3 Effects of Parameter Selection

For conic affinity, the main computational load lies in the non-negative least square approximation. When the number of sample images is large, it is not efficient to apply the full algorithm. Instead, the non-negative least square are only computed for m nearest neighbors of each image. Figures 4.6 show the effects of m on the clustering results for the proposed method with or without k-subspace clustering using the Yale database B and the PIE database. The results show that the method is robust to a wide range of parameter value (i.e., the number of non-negative coefficients in the linear approximation).



(A) (B) (C)

Figure 4.7: Affinity conversion function (A) and the resulting affinity matrices (B) with σ=10 and (C) with σ=100.

In Chapter 3, we discussed the robustness or reproducibility of clustering results. A pilot experiment was conducted on a smaller but similar problem with thorough parameter scanning, then the found parameter was used in the real experiments. Here we present a slightly different approach to show the robustness of the clustering algorithm.

Figures 4.8 and 4.9 show the clustering results with various parameters of the algorithms on three different datasets. The two free parameters, σ in computation of affinities and the number of nearest neighbors (NN), are scaned within a certain range. For the conic affinity, we used the following affinity matrix A' instead of the one defined in Figure 4.2.

$$A'_{ij} \;=\; \exp -\frac{1}{\sigma A_{ij}}$$

The behavior of this function is shown in Figure 4.7. As one can notice from the plots, they present very similar pattern across the dataset, and the minimum error occurs similar points in both σ and NN. This means that one successful parameter set for a (pilot) dataset is likely to give a good clustering result for other similar datasets. Also both algorithms yield reasonably flat regions around the best clustering result, from which one can infer that their clustering performance is robust, and does not require a very fine parameter tuning procedure.

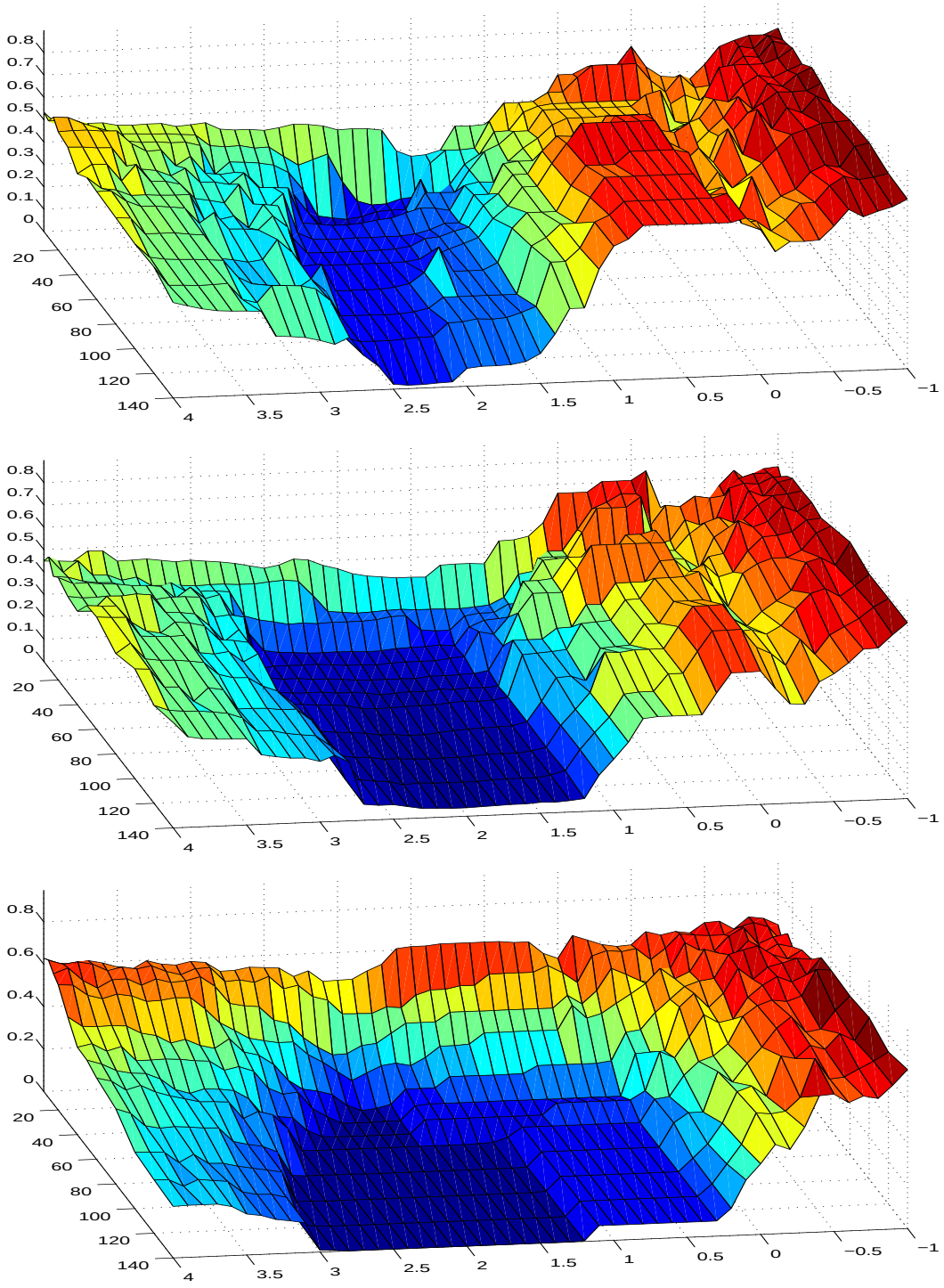


Figure 4.8: Clustering results as sweeping two parameters of the conic affinity with the Yale database B frontal (top), non-frontal (middle) and randomly-selected 20 subjects from PIE66 (bottom). x-axis (long) represents $\log_{10}(\sigma)$, y-axis is the number of nearest neighbors used in affinity computation, and z-axis is the clustering error.

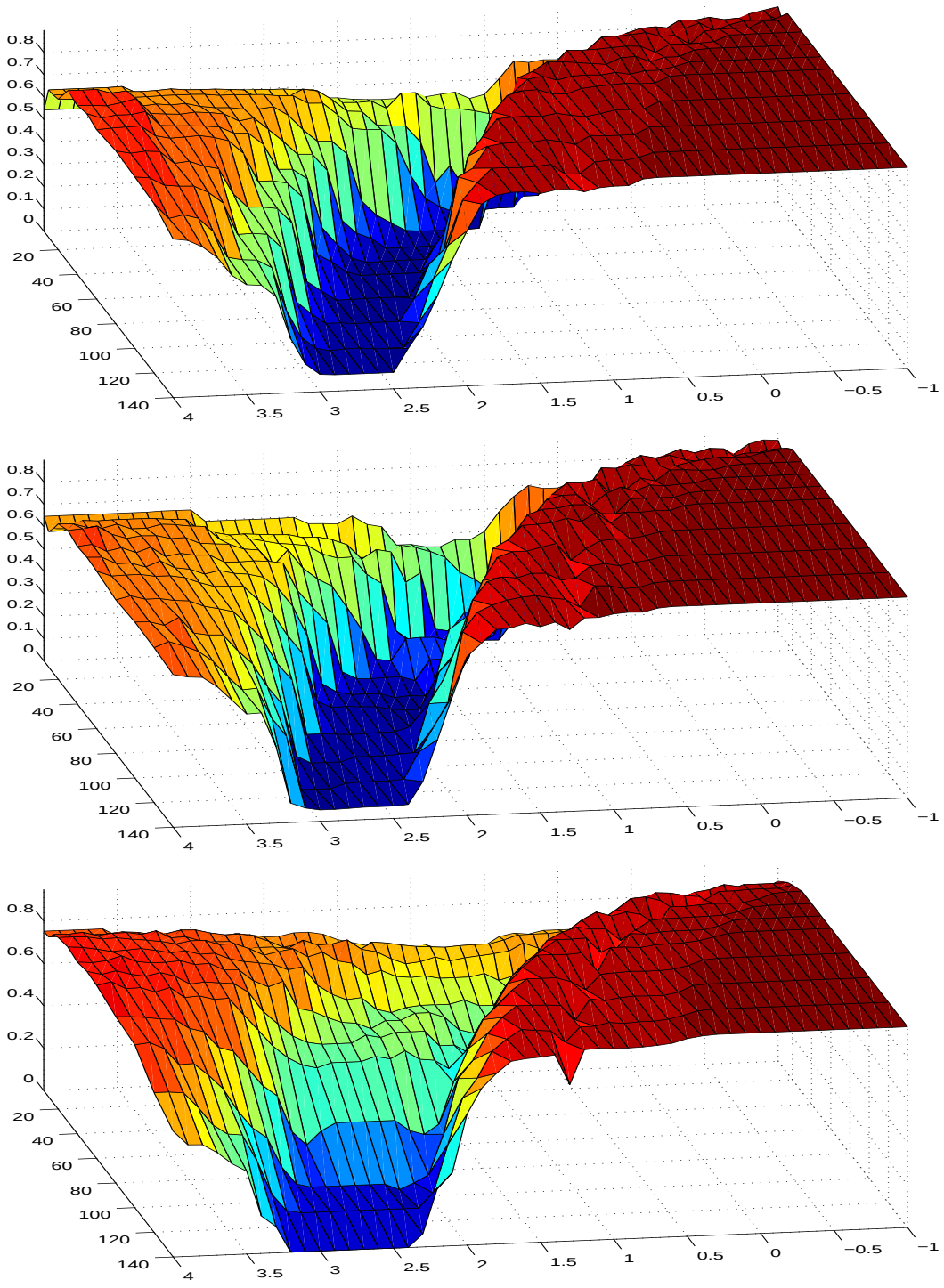


Figure 4.9: Clustering results as sweeping two parameters of the gradient affinity with the Yale database B frontal (top), non-frontal (middle) and randomly-selected 20 subjects from PIE66 (bottom).

4.4 Discussion

We have proposed two appearance-based algorithms for clustering images of 3D objects under varying illumination conditions. We formulate the image clustering problem in a very structured way, and give two similarity measures: *conic affinity* incorporates the global relations among images in the image space, and *gradient affinity* uses the statistical properties of the gradient images of Lambertian objects. We have demonstrated experimentally that the algorithms are very effective with two large data sets. Compared to the previous work, both measures work with the entire image region instead of using feature extraction or pixel statistics. On the other hand, the concept of conic affinity that we proposed here can be applied to other non-vision problem domains where the data points are known to be from some linear or conic structures.

The similarity measures and experimental results complement the earlier results on face recognition [31, 56, 76]. Invariably, these algorithms aim to determine the underlying linear structures using only a few training images. The difficulty is how to effectively use the limited training resource so that the computed linear structures is close to the real one. In this case, the linear structures are hidden among the input images, and the task is to detect them for clustering.

On the other hand, the algorithm proposed in this chapter can be applied to other non-vision problem domains where the data points are known to be from some global linear or conic structures.

Chapter 5

Image Clustering over Viewing Direction Variations

This chapter addresses the problem of clustering images of objects seen from different viewpoints under fixed lighting. That is, given an unlabeled set of images of k objects taken from different viewing directions, we seek an unsupervised algorithm that can group the images into k disjoint subsets such that each subset only contains images of a single object. We formulate this clustering problem under a broad geometric framework. The theme is the interplay between the geometry of appearance manifolds and the symmetry of the 2D affine group. Specifically, we identify three important notions.

- The L^2 distance metric of the image space.

  The algorithm uses the metric to determine a neighborhood structure in the image space for each input image.

- The local linear structure of the appearance manifolds.

  Using local linear structure, comparisons (affinities) between images are computed only among the neighbors.

- The action of the 2D affine transformation in the image space.

  If a pair of images is aligned better after small affine transformation, we use the images after the transformation to compute the affinity value.

Based on these notions, we propose a new image clustering algorithm. Our goal is to identify certain crucial geometric elements, such as the appearance manifold, that are central to the image clustering problem and to formulate a new clustering algorithm accordingly. Specifically, the two main contributions of this work are:

1. The image clustering problem is formulated under a general geometric framework. This framework provides a clear geometric interpretation of our algorithm and comparisons between our work and previous image clustering algorithms.

2. Motivated by geometric considerations, a new image clustering algorithm is introduced.

We have tested this algorithm on two types of image data: images in the Columbia COIL object database and images of human faces. Images of the 3D objects in the COIL database have more variation in surface texture and shape than those of faces, hence local image features can be extracted more reliably from object images [70]. However, for images of human faces, the variations in texture and shape are much more limited, and the features are very similar between individuals, therefore any clustering algorithm employing feature extraction is not expected to do well. We show that our algorithm is capable of producing good clustering results for both types of image data.

5.1 Similarity Measure

In this section, we detail the proposed image clustering algorithm. Schematically, our algorithm is similar to other clustering algorithms proposed previously, e.g., [9, 74] or the illumination clustering algorithm (Chapter 4, [42]). We concentrate on designing a good similarity measure for images with pose variation, and again we use the spectral clustering algorithm to generate the clustering [61].

First, we define the image clustering problem over pose variation. The input to the problem is a collection of unlabeled images $\{I_1, \cdots, I_n\}$ and the desired number of clusters k. We assume that

all images have the same number of pixels N by resizing and cropping, and we obtain a collection of corresponding sample points $\{x_1, \cdots, x_n\}$ in $\mathbb{R}^N$ by rasterizing the images. Our algorithm outputs a cluster assignment for these images $\rho : \{I_1, \cdots, I_n\} \rightarrow \{1, \cdots, k\}$. Two images I_i and I_j belong to the same cluster if and only if $\rho(I_i) = \rho(I_j)$. Ideally a cluster, in our definition, consists of only images of one object. We further assume that the images of a cluster are acquired at different view points but under the same ambient illumination condition.

The problem so formulated is general and without any further information, there is almost no visible structure to base the algorithm on. One obvious structure one can utilize is the ambient distance metric of the image space. The usual L^2 metric or its derivatives (affine-invariant L^2 distance [26] or weighted L^2 distance) are such examples. By considering images as points in $\mathbb{R}^N$, we are naturally led to the notion of appearance manifolds [60]. Accordingly, the input images imply the existence of k sub-manifolds of $\mathbb{R}^N$, $\{M_1, \cdots, M_k\}$ such that two points x_i, x_j belong to the same cluster if and only if $x_i, x_j \in M_s$ for some $1 \leq s \leq k$, with each M_s denoting the appearance manifold of an object. Implicit in the concept of appearance manifolds is the idea of local linearity. That is, if $\{x_1, \cdots, x_l\}$ are points belonging to the same cluster and if they are sufficiently close according to the distance metric, then each point x_i can be well-approximated linearly by its neighbors : $x_i \approx \sum_{j \neq i} a_j x_j$ for some real numbers a_j.

Metric and local linearity are two very general geometric notions, and they do not pertain only to image clustering problems. It is the action of the 2D affine group G that characterizes our problem as an image clustering problem rather than a general data clustering problem. (henceforth in this chapter, except at a few places, G will invariably denote the 2D affine group.) If $\{x_1, \cdots, x_n\}$ were data of a different sort, e.g., data from a meteological or high energy physics experiment, there will not be an explicit action of G. It is precisely because the 2D nature of the images and the way we rasterize the image to form points in $\mathbb{R}^N$, that we can explicitly calculate the action of G given a sample point x. In particular, each appearance manifold M_i is invariant under G, i,e, if $x \in M_i$ then $\gamma(x) \in M_i$ for each $\gamma \in G$. In this sense, the clustering problem acquires a symmetry played by the 2D affine group. (Strictly speaking, the symmetry group will depend on what type of

imaging model is used for the problem. In general, it will be a subgroup of G rather than G itself.)

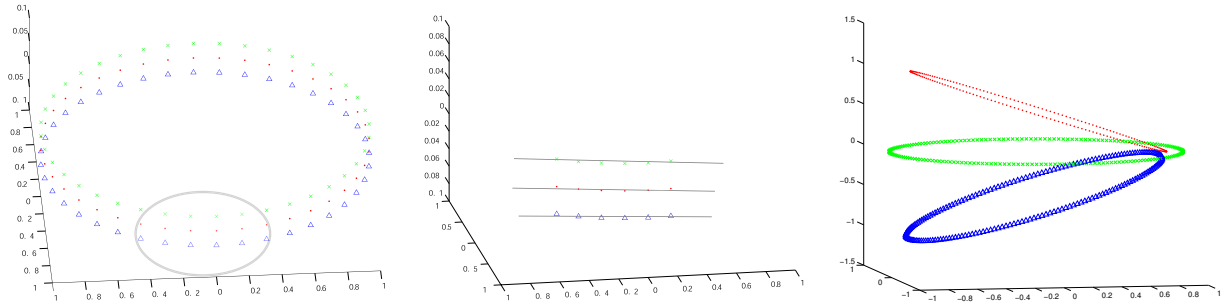
In summary, we have identified three important elements to this image clustering problem. First, there is the ambient L^2 (and its derivatives) metric of the image space. Second, each cluster has local linear structure. The metric and local linearity are the only two geometric structures that we can utilize in designing the algorithm. The third element is the affine symmetry of the problem.

Our challenge is to design a clustering algorithm that takes into account these three elements. In a very general outline, what is needed is to design metric and local linear structure that are both invariant under the affine group G *and* to seek an interesting and effective coupling between the metric and linear structure, which are two rather disparate geometric notions. Surprisingly, using only these three very general structures, we can formulate a clustering algorithm which will be demonstrated to be effective for a variety of image clustering problems. The algorithm is compact and purely computational.

5.1.1 Metric Structure

Since the input images are considered as a collection of points in $\mathbb{R}^N$, the usual L^2-distance metric and its derivatives offer the simplest affinity measures between a pair of data points. However, since the clusters form manifolds in $\mathbb{R}^N$, the images are not expected to localize in a compact region of $\mathbb{R}^N$ independently of other clusters. This observation can be supported by the fact that the Euclidean distance between two face images of different identities acquired at the same pose is almost always smaller than the Euclidean distance of two images of the same identity but acquired at different poses [36, 66, 67]. Two analogous situations in a 3D feature space are depicted in Figure 5.1. They clearly demonstrate that if the metric information is used for defining affinity, then "medium" and "long-distance" comparisons are usually erroneous.

However, Figure 5.1.B suggests one good way of using the metric is not to use it directly for comparison. Instead, we can use the metric to pick data points for which the comparisons will be made. In particular, for each point x, the metric defines a neighborhood and in this neighborhood, non-metrical information can be exploited to do the comparison (i.e., defining affinity). In this way,



(A) (B) (C)

Figure 5.1: (A) Three parallel circles. The points on each circle are uniformly sampled and the distance between adjacent circles is slightly smaller than the distance between two neighboring points on the same circle. (B) A "magnified" view of a neighborhood. (C) The top and bottom circles are rotated by $\pm 30°$.

the metric defines a collection of local affinity estimation problems, and the affinities computed in these local settings will then be put into the global affinity matrix to provide a final clustering result.

5.1.2 Local Linear Structure (LLS)

Figure 5.1 shows two examples which are unlikely to be clustered correctly using the metric information alone. Figure 5.1.A is a good example. The data collection contains points sampled uniformly from three circles in $\mathbb{R}^3$. The distance between adjacent circles are slightly smaller than the distance between two neighboring points on the same circle. To the best of our effort, we can not correctly cluster the data into three circles using only metric information. The point of course is that the manifold structure of the circles must be taken into consideration. One possible way to use the manifold structure is to compute the "tangent space" at each sample point using Principal Component Analysis in a neighborhood of the sample point, as in [9]. This approach can correctly cluster Figure 5.1.A but unlikely to cluster Figure 5.1.C correctly. This is mainly because the local linear estimate using PCA becomes unstable in the region when the circles come into close contact with each other.

Instead of working with tangents, we shift our focus slightly to consider the secant approxi-

mation of a sample point by its neighbors, see Figure 5.2.A. For a smooth 2D curve, each point x can be approximated well by a point on the secant chord formed by two of its sufficiently close neighbors y_1, y_2, i.e., $x \approx a_1 y_1 + a_2 y_2$ with a_1, a_2 non-negative and $a_1 + a_2 = 1$. This can be generalized immediately to higher dimension: for a point x and its neighbors, $\{y_1, \cdots, y_K\}$, we can try to compute a set of non-negative coefficients ω_i which is the solution to the following optimization problem:

$$\min \left\| x - \sum_{i=1}^{K} \omega_i y_i \right\|_{L^2}^2 \tag{5.1}$$

with the constraint that $\omega_i \geq 0$ and $\sum_{i=1}^{K} \omega_i = 1$. Assuming $\{y_1, \cdots, y_K\}$ are linearly independent (in the image space $\mathbb{R}^N$, this is almost always true since $K \ll N$.), then the coefficients ω_i are unique. Figure 5.2.B illustrates that the magnitude of the coefficients ω_i can be used as an affinity measure locally to detect the presence of any linear structure. This can be easily verified by considering the Barycentric coordinates of a point in a triangle. A coordinate value is large (close to one) when the point is close to the corresponding vertex of the triangle, and close to zero when it is far from the vertex. That is, a large magnitude of ω_i indicates the possibility that y_i and x share a common local linear structure.

Applying the idea we have outlined so far, a simple data clustering algorithm can be designed:

1. Compute ω_i for each sample point using its K-nearest neighbors provided by the metric.

2. Form an affinity matrix using ω_i, make it a symmetric matrix and apply the spectral clustering algorithm.

We can cluster all the examples above correctly. To our best effort, we can't find any simple and straightforward algorithm, based on the more traditional clustering techniques such as the k-means and connected component analysis, etc., that can successfully cluster all of these examples.

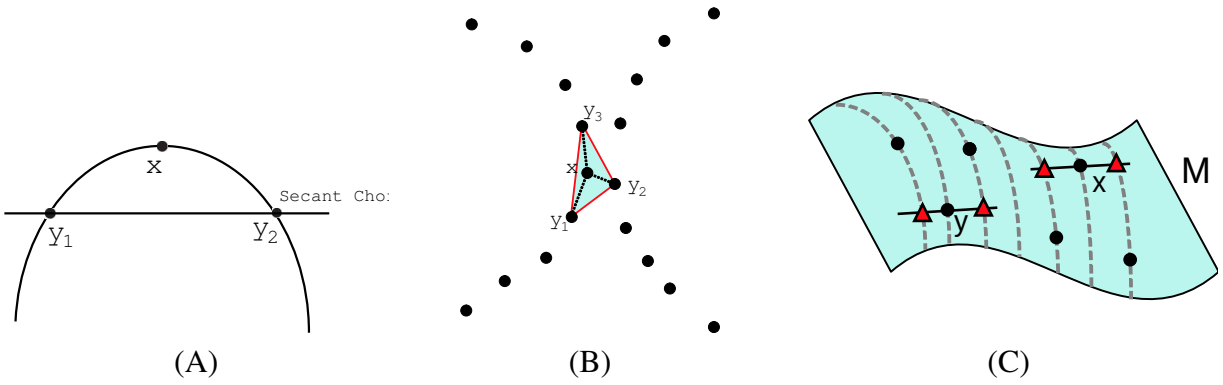


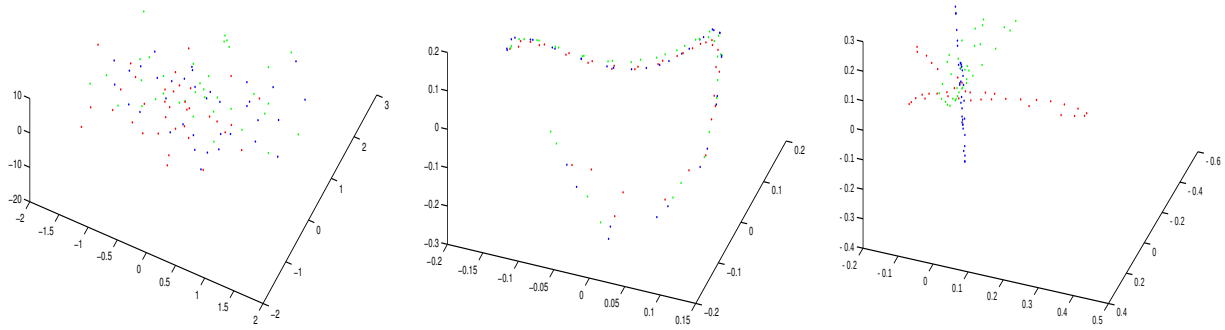
Figure 5.2: (A) The secant chord approximation of a point on a smooth curve by its neighbors. (B) Two semi-circles. There are three possible secant chord approximations of x by the three sides of the shaded triangle. (C) The shaded surface denotes the appearance manifold M and the dashed lines are the trajectories of affine-transformed versions of each image. The solid circles denote the sample points. In order to construct the local linear structure, we move the sample points close to the reference point (x or y) using affine transformation to produce "virtual" samples denoted by the triangles.

5.1.3 Affine Transformation for Small Pose Variation

As we mentioned earlier, changes in images caused by small 3D pose variations can be well modeled by 2D affine transformations. In the image space, this action results in changes in the locations of the image points in the dataset. Unlike general data clustering, we need to measure similarities between these 'floating' data points. The task now is to put both the metric and local linear structure into an affine invariant setting (as best as we can). Affine invariant L^2 metric and many of its variants have been studied before in the literature [25,26,77], etc. Our effort is to propose a method for defining local linear structures that are affine invariant; in particular, we want to reformulate Equation 5.1 in an affine-invariant way.

To compute the local linear structures from the images $\{I_1, \ldots, I_N\}$, we determine K nearest neighbors $\{y_1, \ldots, y_K\}$ of an image x. It is desirable to use the "two-wided distance" in computing the distance between two images x and y, defined as

$$\tilde{d}_G(x, y) = \min_{\gamma_x, \gamma_y \in G} \|\gamma_x(x) - \gamma_y(y)\|_{L^2}^2$$



(A) (B) (C)

Figure 5.3: Embedding Results for the three cars in COIL20 (see next section). (A) PCA projection. (B) A direct application of the LLE embedding [69]. (C) Using our local linear structure.

where G is the set of all possible affine transformation. However, in addition that the computation of the distance is not easy and straightforward to be implemented, the distance can be arbitrarily small or even zero for any two images by applying the transformation which maps the entire image to a single point.

In the actual implementation, we use the "one-sided distance" [26]. For each input sample y, the "one-sided distance" is defined as

$$d_G(x, y) \;=\; \min_{\gamma \in G} \Big\{ \min \big\{ \, \|x - \gamma(y)\|_{L^2}^2, \, \|y - \gamma(x)\|_{L^2}^2 \, \big\} \Big\}. \tag{5.2}$$

Although $d_G(x, y)$ is not a metric, it still allows us to define the K-nearest neighbors of x. The K neighbors $\{y'_1, \cdots, y'_K\}$ of x above are just $\{\gamma_1(y_1), \cdots, \gamma_K(y_K)\}$ with each γ_i minimizing the one-sided distance between x and y_i.

Figure 5.3 shows what the affine-invariant abstract space might look like. The computed coefficients ω_i fit very well with the local linear embedding (LLE) introduced in [69]. In fact, they can be directly used for computing the embedding, and the result is shown in Figure 5.3.A. The difference between the embedding result for a typically LLE application and the embedding result shown here is that typically for an LLE embedding, the data already lives in some high dimensional

Euclidean space, and the local linear relation is directly computed from the data. In contrast, for this embedding, the abstract space does not apriori belong in any Euclidean space, and its existence is only implied by the input data. Its local linear structure is computed using both the input data and the affine measure (Equation 5.2).

Furthermore, there are two important things to note in Figure 5.3. First, compared with other results (PCA projection for example), this embedding result clearly shows a larger degree of separation between clusters than other embedding results. Second, the separation still does not guarantee an easy clustering task. In fact, the situation is similar to the three slanted circles shown in Figure 5.1.C in that parts of the clusters come into contact with other clusters in a rather complicated manner. This result suggests that a direct application of the metric information for clustering is probably not sufficient.

Putting together the discussions above, we have our image clustering algorithm, which is summarized in Figure 5.4. The outline of the algorithm is very simple: for each data point, we use the metric to choose the neighbors. In the neighborhood defined by the data point and the chosen neighbors, we exploit the local linear structures of the clusters to define an affinity (local affinity) by using the coefficients ω_i (Equation 5.1). The tricky part is to do this in a way that is compatible with the action of the 2D affine group. All the local affinities are then put together in a global affinity matrix to yield the final clustering result.

5.2 Comparison with Previous Work

In this section, we compare our algorithm with some of the well-known image clustering algorithms in the literature. Needless to say, the 2D affine group has a long history in the computer vision literature. In particular, intensive effort has been focused on studying (quasi-)affine invariant metric such as the tangent distance e.g., [28, 77]. For image clustering, affine invariant metric has made its appearance in the work of Fitzgibbon and Zisserman [25, 26]. Most of the effort in these two papers has been focused on designing an affine invariant metric that will be effective for

1. **Inputs**
   A collection of unlabeled images $\{I_1, \cdots, I_n\}$. Considered the images as data points $\{x_1, \cdots, x_n\}$ in the image space $\mathbb{R}^N$, and the number of clusters k.

2. **Use Metric to Choose Neighbors**
   For each data point x, compute a set of K nearest neighbors using the distance measure $d_G(x, y)$ defined above.

3. **Use local linear structure**
   For each x and its K-neighbors $\{x_1, \cdots, x_K\}$ determined in the previous step, let $\{y_1, \cdots, y_K\}$ be the points in $\mathbb{R}^s$ such that $y_i = \gamma(x_i)$ for some $\gamma \in G$ and y_i minimizes the distances between x and all points on the orbit of G through x_i. Using y_i's to linearly approximate x by determining a collection of K non-negative real numbers ω_i that minimize the objective function

$$\left\| x - \sum_{i=1}^{K} \omega_i y_i \right\|_{L^2}^2, \quad \text{with the constraint that } w_i \geq 0 \text{ and } \sum_{i=1}^{K} w_i = 1$$

4. **Use ω_i as the affinity measure**
   Define an affinity measure d_Ω between two data points x_i and x_j: $d_\Omega(x_i, x_j) = \min(1/\omega_{ij}, 1/\omega_{ji})$, where ω_{ij} is the coefficient computed in the previous step for x_i. If x_j is not among the K-neighbors of x_i, ω_{ij} is set to 0. Apply the spectral clustering algorithm (e.g., [61]) using this affinity to yield the final clustering result.

Figure 5.4: The image clustering algorithm over pose variations.

clustering. In the language of the quotient space, they are doing clustering on M/G using metric information alone. Our algorithm also uses the metric information in M/G but it also explicitly tries to cluster "manifolds" instead of "points". Although good clustering results can be obtained by considering metric alone, we believe that by incorporating both the metric and local linearity, it offers 1) a more effective clustering algorithm and 2) a more complete geometric description of the clustering algorithm.

Another well-known image clustering algorithm that explicitly uses the concept of the appearance manifold is Basri et al.'s work [9]. However, there are two major differences between our work and theirs. First, the affine symmetry is absent in their work. One of the main themes in this work is that the action of the 2D affine group is of central importance in formulating any image clustering problem. Second, there is an important difference between our concept of local linearity and theirs. In their work, the concept of local linearity is embodied in the idea of tangent space of the appearance manifold; therefore, PCA is used to estimate local linear subspaces. In contrast, our concept of local linearity is on how best the "neighbors" can linearly approximate a given sample point, and it is formulated through Equation 5.1. This concept of local linearity also allows non-geometric interpretation in terms of image comparisons using parts of objects as in [55]; however, it is not clear if there is a non-geometric interpretation of the tangent spaces used in [9].

Frigui et. al. and LeSaux & Boujemaa [29, 70] are two other interesting and related papers on image clustering. Their approaches and ours are fundamentally different in that our algorithm is completely image-based while their algorithms focus on extracting salient image features and incorporating more sophisticated machine learning techniques for clustering. However, comparisons between their experimental results and ours will be made in the next section.

5.3 Experiments

In this section, we report our experimental results. Our image clustering algorithm, as detailed in Figure 5.4, has been implemented in MATLAB. Two different types of image data were used to test

the algorithm, images of 3D objects and images of human faces. Substantial variations in appearances are observed in all image datasets. The main difference between these two types of datasets is the variation in surface texture. For the former type, the surface texture varies greatly and local image features (such as corners) can be more reliably extracted. Human faces, on the other hand, have much limited variation in surface texture and local image features become less useful. Traditionally, these two different types of image data were attacked separately using feature-based methods (e.g., [70]) and appearance-based methods (e.g., [9, 58]), respectively. However, the results below show that our algorithm is capable of obtaining good clustering results for both types of images.

Except for the affine-invariant metric $d_G(x, y)$, the implementation is straightforward and it follows closely the steps outlined in Figure 5.4. Given two images, I_1 and I_2, $d_G(I_1, I_2)$ is computed as follows. First, we assume a Gaussian distribution p on 2D affine group centered at the identical affine warp. Since we only consider small affine corrections, p can be expressed in a local coordinate system centered at the identity by expressing each (small) affine transformation in terms of the usual six parameters (a 2×2 matrix plus 2 translations). Using these six parameters, p is a Gaussian distribution with diagonal covariance matrix. Next, we determine an affine transformation γ such that it minimizes the function

$$E(\gamma) \;=\; \min\left\{\, d_{L^2}(\gamma(I_1), I_2),\, d_{L^2}(I_1, \gamma(I_2)) \,\right\}$$

where d_{L^2} is the usual L^2 distance metric between two images. γ can be found using gradient descent based methods [6, 26, 38]. $d_G(I_1, I_2)$ is then defined as the sum $E(\gamma) - \log p(\gamma)$. The reason for incorporating the Gaussian $p(\gamma)$ is to penalize "over-corrections" by large affine transformations [26].

(A)

(B)

(C)

Figure 5.5: (A) Representative images of objects in COIL20 (B) Representative images of objects in COIL100 (C) The ten vehicles in VEH10.2

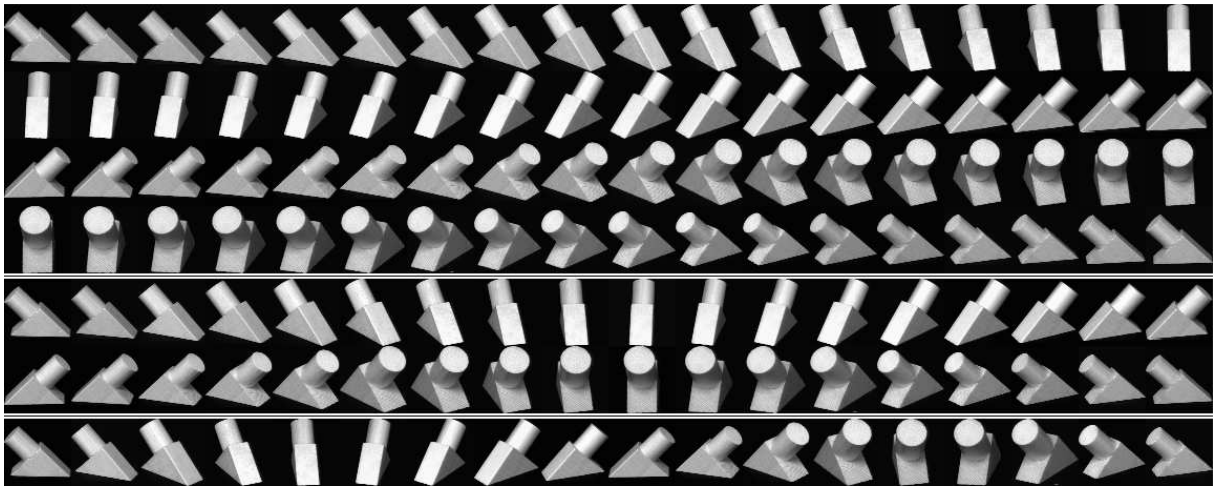


Figure 5.6: Sampling frequency. First 4 rows are images of one object in the original COIL dataset. Next 2 rows are subsampled version with with a factor of 2, and the last row is 4-subsampled version.

(A)

(B)

Figure 5.7: (A) Individuals in the FACE10 database. (B) Pose variation in FACE10. Images are cropped manually from video sequences of people turning their heads around.

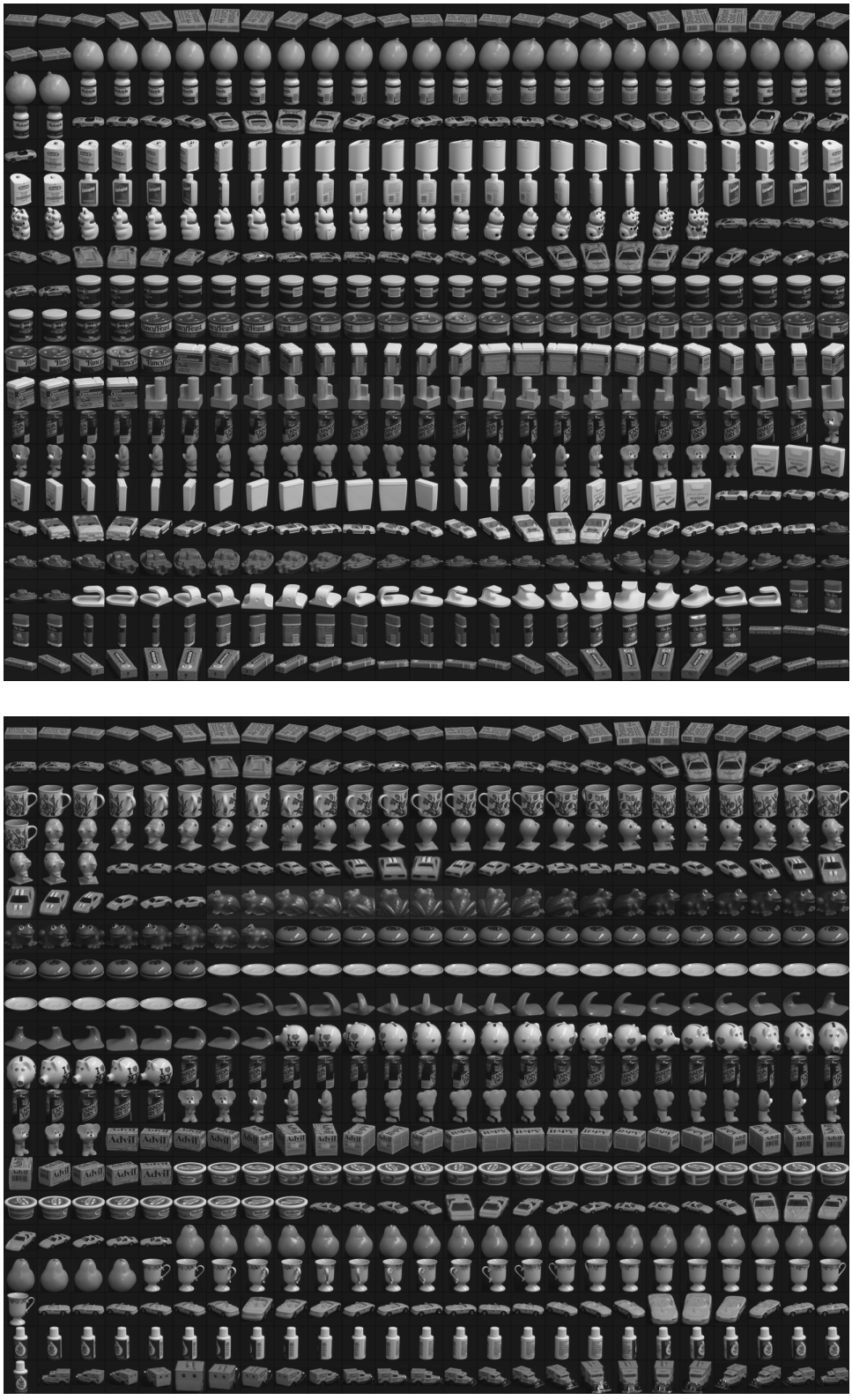


Figure 5.8: Two of 10 datasets randomly selected from the COIL100.2 dataset.

5.3.1 Datasets

In this subsection, we fix the notations for various image datasets we used in the experiments and give brief descriptions of the datasets. For images of 3D objects, we use the COIL datasets from Columbia, which are popular datasets for validating object recognition algorithms. There are two COIL datasets, COIL20 and COIL100 (Figure 5.5.A and 5.5.B). They contain 20 and 100 objects, respectively. For both datasets, the images of each object were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. Since this sampling is quite dense, we "sub-sampled" the image collections to make clustering problem more interesting. We let COIL20.2 denote the collection of images obtained from COIL20 by sub-sampling it with a factor of 2 (Figure 5.6). So COIL20.2 contains the same number of objects as the original COIL20 but with half as many images per object. Similarly, COIL20.4 denotes the collection obtained from COIL20 by sub-sampling it with a factor of 4 and so on. From COIL100.2, we placed all vehicle images in this collection together to form a new dataset, VEH10.2 (Figure 5.5.C). The images of these vehicles have similar appearances and therefore, they offer a challenging dataset to test our algorithm. For images of human faces, we collected video sequences of ten individuals to form ten image sequences with each sequence containing 50 images (Figure 5.7). Pose variation in this collection is quite large and because of the differences in individual motion, the image sequences do not have uniform variation in pose. This dataset will be denoted as FACE10.

The COIL datasets have all 360 degree views of each object with regular sampling viewing directions. To simulate more realistic scenario, we built 10 subsets (COIL20s) with 500 images from the images of randomly selected 20 objects in the COIL100.2 dataset. Images in these datasets no longer are regularly placed in viewing directions, nor have the same number of images per each object. Figure 5.8 shows two of COIL20s datasets as examples.

	FACE10	COIL20	COIL20.2	COIL20.4	VEH10.2	COIL100.2	COIL100.4
Error	0.00%	0.00%	0.00%	15.56%	11.11%	20.69%	34.89%
K	10	8	30	15	3	6	10

Table 5.1: Clustering results of our algorithm on the date sets.

5.3.2 Results

The experimental results are reported in Table 5.1. As is clear from Table 5.1, our algorithm produces good clustering results for all datasets except COIL100.4. The algorithm's performance on COIL100 is not surprising considering that there are 100 objects in COIL100.4 and the images are rather sparsely sampled (every 20 degrees). Error rates are calculated as the ratio of the number of misclustered images over the number of images. For each cluster emerging from the clustering result, we try to match it with the known clusters (ground-truth). Once the one-to-one map between the new clusters and known clusters is computed, the error ratio can be calculated accordingly. For instance, a random assignment of a collection of N clusters of equal size will on average produce an error rate of $\frac{N-1}{N}$ according to our definition. The error rates are shown together with the parameter K which defines the size of the local neighborhoods. We also mention that there are clustering results on COIL20 database reported in [70]. We can not translate their definition of errors into ours. However, they do report non-zero error rate while our clustering algorithm achieves a perfect clustering result for the COIL20 dataset.

Table 5.2 shows the clustering result on the COIL20s datasets. Although the dataset is built in a somewhat challenging way, the algorithm still performs well for most of the datasets. As we will discuss later, the parameters which give the best clustering result are mostly similar and do not vary much.

In Figure 5.9, we illustrate several results of our local linear estimates, i.e., the ω_i. Although the K-nearest neighbors of an image generally contain images of other objects, in each case, ω_i correctly pick out the right images to form strong affinities.

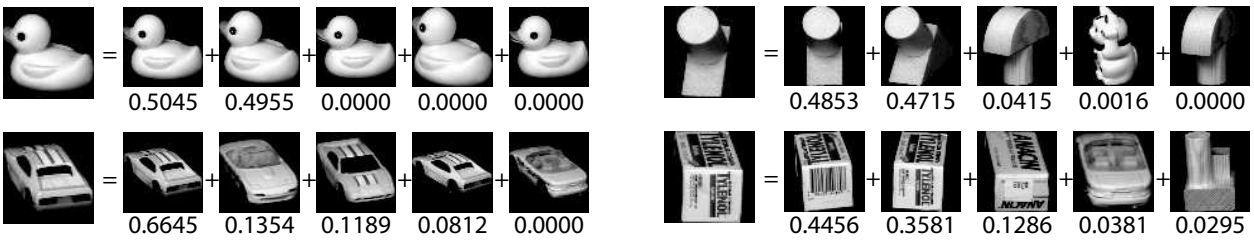


Figure 5.9: Images, their neighbors and the local linear structure, ω_i's.

COIL20s	1	2	3	4	5	6	7	8	9	10	avg
Error (%)	15.60	4.60	9.60	16.60	8.00	5.00	14.60	14.00	0.20	6.80	12.40
K	50	50	50	50	10	30	50	50	50	40	40
$\log_{10}(\sigma)$	1.60	1.20	2.00	1.20	2.20	1.60	1.80	2.20	1.50	1.90	1.90

Table 5.2: Clustering result of 10 COIL20s datasets.

5.3.3 Comparison with other clustering algorithms

Table 5.3 lists the result of comparing (on four different datasets) our algorithm with some standard off-the-shelf algorithms. First, two standard clustering algorithms, k-means and spectral clustering algorithm [61] with the usual L^2-distance metric, are compared with our results. It clearly demonstrates that direct L^2 comparisons without affine-invariance are not sufficient. Next, we incorporate affine-invariance but without using local comparisons (Affine+Spectral). This is the "one-sided" distance measure [26] and again, it is still not able to produce good clustering results. Next, we show that by incorporating local linear structure in the algorithm, it does indeed enhance the performance of the clustering algorithm. Note that in our framework, once a neighborhood structure has been determined, we exploit the local linear structure to cluster points in the neighborhood. To show that this is indeed effective and necessary, we replace this step of our algorithm with direct metric comparisons. That is, we are computing local affinities based purely on the "one-sided" distance measure (Affine+K-NN+Spectral). We expect that our algorithm will be an improvement over this method because of our use of non-metrical information, and the results do indeed corroborate the claim.

Algorithms	Datasets			
	COIL20.2	COIL20.4	FACE10	VEH10.2
Our algorithm	0.00%	15.56%	0.00%	11.11%
Affine+K-NN+Spectral	7.36%	21.11%	13.00%	27.50%
Affine+Spectral	10.14%	25.83%	22.00%	40.00%
Euclidean+Spectral	35.14%	33.06%	25.60%	61.67%
Euclidean+K-means	39.58%	48.06%	46.00%	74.44%

Table 5.3: Comparison with other clustering algorithms

5.3.4 Effects of Parameter Selection

Similar to the discussion in Chapter 4, we present the result of scanning parameters of the proposed algorithm. The algorithm has two free parameter, σ for the affinity computation and K, the number of neighbors considered in the weight estimation process. Figure 5.10 shows three graphs of COIL20.2, COIL20.4 and average errors of 10 COIL20s datasets.

Also we show how the clustering algorithm performs with each COIL20s dataset in Figure 5.11. One can easily see that the result shows a very stable pattern over the three dataset, and also 'good parameters' can be easily identified from the figure.

5.4 Discussions

In this chapter, an image clustering algorithm over pose variation is proposed and its effectiveness is demonstrated by the clustering results of various datasets of 3D objects undergoing large pose variation. We formulated the affinity measuring problem in terms of the geometry of appearance manifolds and the symmetry of the 2D affine group. Specifically, we identified three important notions for image clustering: the L^2 distance metric of the image space, the local linear structure of the appearance manifolds, and the action of the 2D affine group in the image space. Briefly the algorithm uses the metric to determine a neighborhood structure in the image space for each input image considering the affine motion between images. Using local linear structure, similarities between images are computed only among the neighbors.

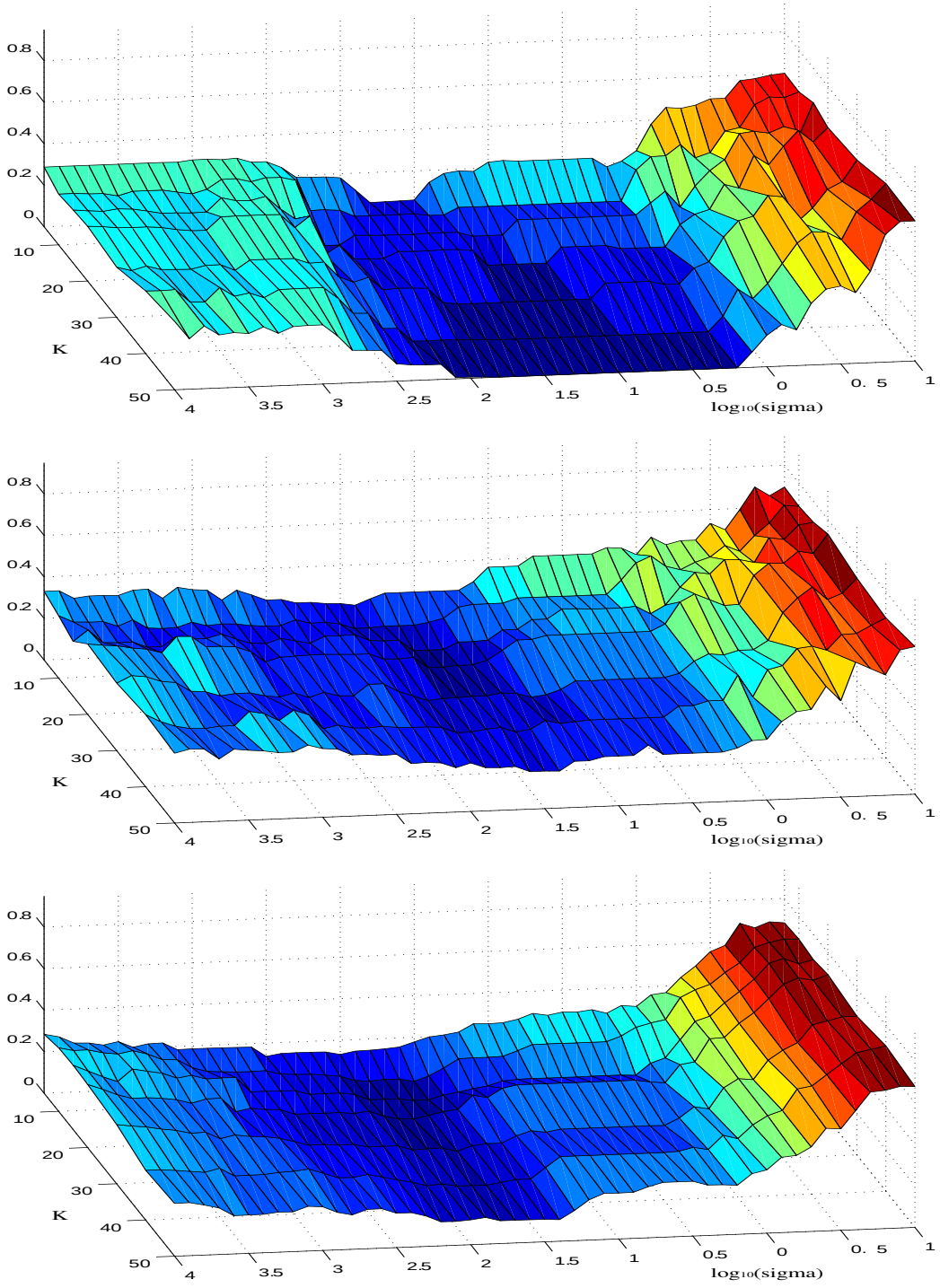


Figure 5.10: LLS clustering results as sweeping two parameters with the COIL20.2 (top), COIL20.4 (middle) and the average result of 10 COIL20s datasets which is built with the images of randomly-selected 20 objects in COIL100.2 dataset (bottom). x-axis (long) represents $\log_{10}(\sigma)$, y-axis is the number of nearest neighbors used in affinity computation, and z-axis is the clustering error.

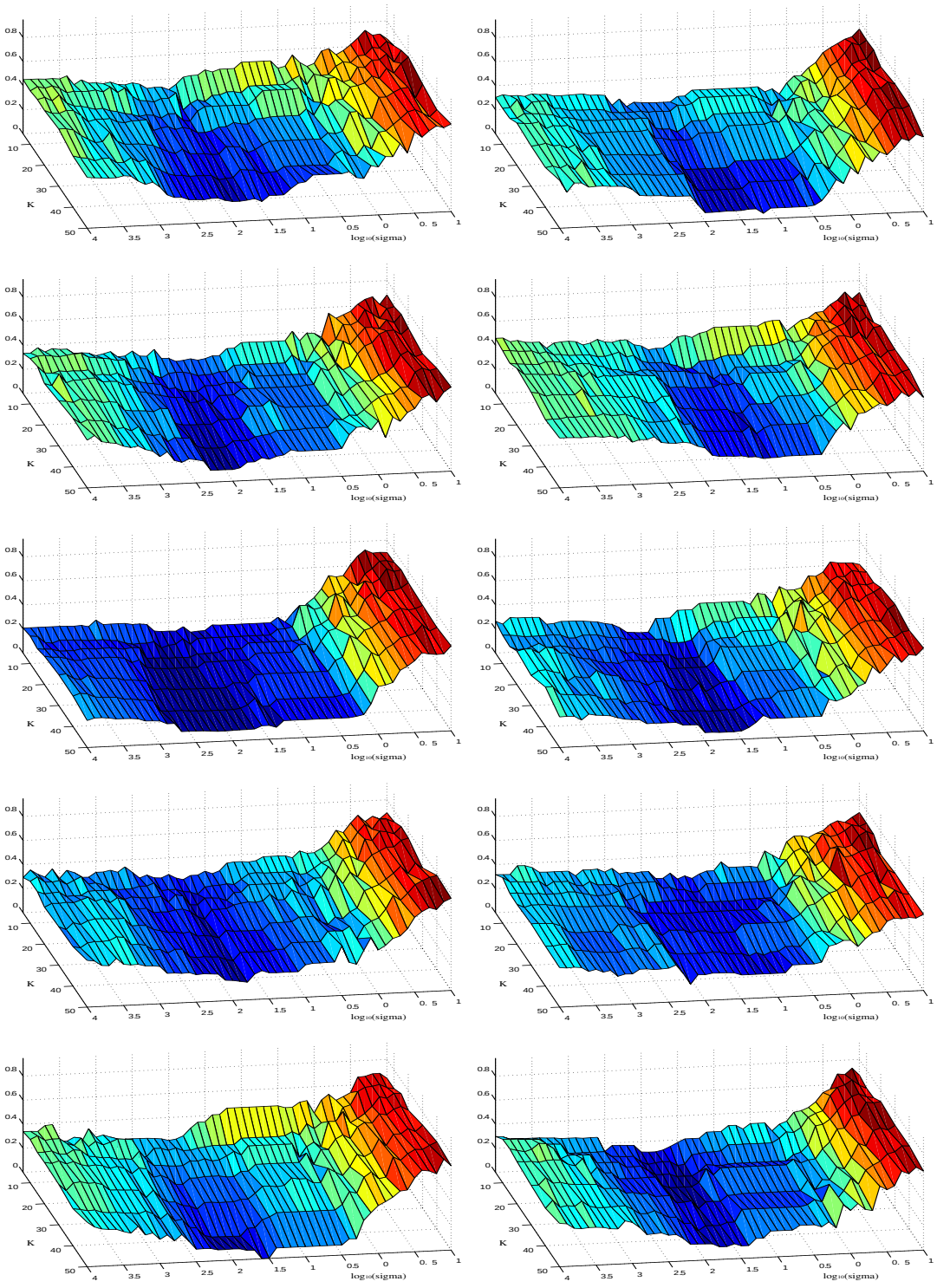


Figure 5.11: LLS clustering results of 10 COIL20s datasets.

One obvious limitation of the proposed algorithm is that it does not have an explicit model of the illumination effect. However, in Chapter 4 we have demonstrated that it is possible to handle large amount of illumination variation using global linear structures or local image gradient information.

Chapter 6

Clustering Images with Lighting and Pose Variations

In Chapter 4 and 5, lighting and pose variation are independently studied and a few algorithms are proposed for clustering.. In most real world situation, these two kinds of variation occur simultaneously. Even when an objectis rotated on a turn table and the lighting does not change, the illumination condition with respect to the object changes as the object changes its pose. Thus to cluster real images or video sequences, an algorithm is needed that can handle both changes together.

For situations covered in Chapter 4 and 5, algorithms can be evaluated on many publicly available databases or one can relatively easily collect one's own dataset using still or video cameras. When both external imaging condition vary, it is more difficult and burdensome to build an image database. To the best of my knowledge there is no public dataset of objects besides faces whose images are taken under controlled lighting and pose variation. For faces, there are Yale face database B and CMU PIE database with such variation, but the available poses of the datasets are rather limited in amount and diversity. By personal communication, we received a dataset from Athinodoros Georghiades at Yale University, and a detailed description will be given in the next section.

6.1 Algorithmic Approach

Our approaches are mainly in two direction: extending the Local Linear Structure (LLS) algorithm to handle illumination changes and extending the gradient affinity algorithm to handle pose variations.

6.1.1 Extension to LLS

For some degree of lighting variation, especially when there is significant ambient light, the gradient-based affine image alignment algorithm empirically works well. Given that neighboring images are aligned correctly, the LLS algorithm becomes identical to the conic affinity measure introduced in Chapter 4: they both compute non-negative linear coefficients of (aligned) neighbor images to the target image. When the affine alignment is roughly correct and there are a few images of the same object in the neighbor image set, the LLS/conic affinity must assign the right weights to those images.

To make sure the LLS/conic affinity work, we need to check

1. if the neighbor images are aligned correctly to the target image, and

2. if the estimated weights correctly reflect the object identity relations.

The second issue can be justified from the result of the previous chapters about clustering images over illumintion and pose, so here we concentrate on the first issue: finding a good alignment between images under possible lighting variations and (small) pose variations.

In this chapter, we consider two possible extensions of the affine alignment algorithm. The affine-alpha alignment finds the affine parameter and α value which minimizes the following error function for two images I_0 and I:

$$E_\alpha \;=\; \sum_{\mathbf{x}} \left\| I(w(\mathbf{x};\boldsymbol{\theta})) - \alpha I_0(\mathbf{x}) \right\|^2 .$$

The additional α allows for the algorithm to handle global intensity changes. We call this alignment algorithm as *LLS*α.

Another approach is to substitute the gradient affinity measure for usual L_2 norm in choosing neighboring images and computing affine alighment between images. This method is referred as *LLSgrad*. In Section 6.2, we discuss the effectiveness of these two extensions.

6.1.2 Extension to Gradient Affinity

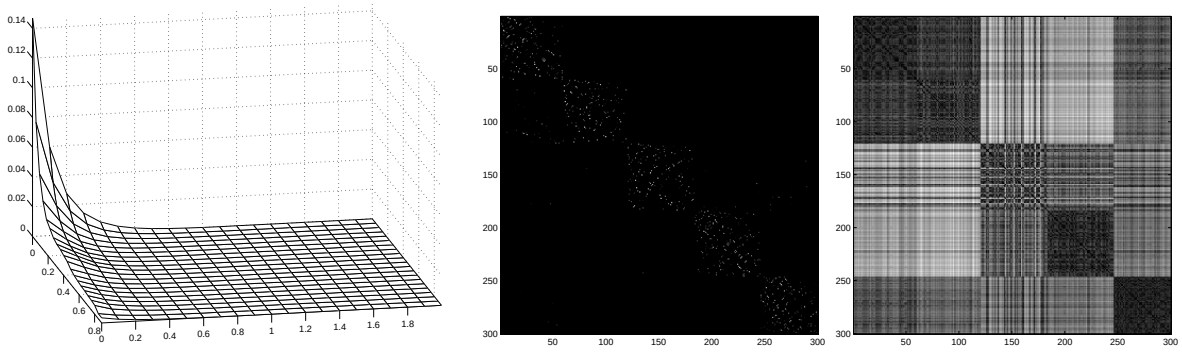
Compared that the LLS/conic affinity works with multiple neighboring images, the gradient affinity can be computed from a pair of images. Once you have one images aligned correctly to the other image, the original gradient affinity can be applied to fill the affinity matrix.

It is unreasonable to assume that all pair of images can be affine aligned to each other, since the affine approximation only works for small 3D pose changes. In extending the gradient affinity, only nearby images can be used, so the resulting affinity matrix is very sparse.

In addition to affine alignment, the histogram of image gradient vectors can be used to evaluate the similarity between images. Since it is hard to use the exact functional form of Chen et al.'s measure [17], the approximate version is used in Chapter 4. Here we use a 2D histogram of angular difference and magnitude difference of the image gradient vectors at each pixel in two images. The histogram is built from the Yale B face database, and it is shown in Figure 6.1. This approach is called *LGAcb2*. Also the histgram can be build in 3D as two gradient magnitudes and their angular difference. This 3D histogram is built in the same way as the 2D version with Yale B Face database. The clustering algorithm using the 3D histogram is called *LGAcb3*.

6.1.3 Geodesic Expansion of Affinities

Intrinsically the affinity matrices for pose and light variation are sparse. Sparsity helps to reduce the amount of computation, but it also means there is not much information in the matrix (compared to a full matrix). The geodesic distance is a very natural measure of distance between data points if



(a) (b) (c)

Figure 6.1: (a) 2D histograms of gradient angular difference (x-axis) and magnitude difference (y-axis) collected from Yale B face database. (b) An affinity matrix after running the LLS algorithm on one of YCOIL dataset, and (c) the geodesic distance matrix computed from it.

it is known that the data points are from a manifold. The Isomap algorithm [78] uses the geodesic distance and MDS to find a low-dimensional embedding of data points which reside in a high dimensional data space. A similar approach can be applied in our case to fill in empty elements in the sparse affinity matrix (Figure 6.1). To use the geodesic expansion, one unclear but important issue is how to convert an affinity to a distance. When the affinity is defined from some distance between items, the distances can be used as the input to the geodesic expansion before converting to affnities (e.g., the gradient affinity in Chaper 4). However, like the LLS or conic affinity, some affinity measures do not have a 'natural' distance interpretation. In these cases, we can take some arbitrary but reasonable conversion from affinty to distance. In this experiment, we take the inverse of each affinity value (1/affinity) and treat it as a distance. Since the range of valid affinity values is between 0 and 1, the inverse varies between one and infinity and more similar items have a shorter distance. After geodesic expansion, each geodesic distance is converted to an affinity by inverting it.

6.2 Experiments

6.2.1 YCOIL Dataset

The YCOIL dataset is collected by Athinodoros Georghiades at Yale university. The dataset consists of 18 objects which are chosen from the COIL100 database. Each object was put on a turn table and rotated by 15 degree (24 poses), and per each pose 64 images were captured with flashes. The flash configuration is same as the Yale B face database [31], and depending on the angle between the camera viewing direction and the light direction, 5 subsets are defined so that subset 1 contains images with lighting directions close to the camera (frontal light) and subset 5 are directions that are almost perpendicular to the viewing direction. Subset 4 and 5 are very extreme lights, so in this experiment these two subsets were not used.

Since all the illumination is directional and there is no ambient light, the image of some object with planar surfaces are very harsh. To make the dataset more realistic, we simulated ambient lighting by taking the average over the lighting variation of all images in each pose of each object, then mixed it with the original directionally illuminated images by taking a linear combination of the ambient image and the original image ($\tilde{I} = \alpha I_{amb} + (1 - \alpha) I_{org}$).

The images in the original dataset are 640x480 grayscale. To segment the image each image is tightly cropped after adding the ambient lighting, and resized to 64x64 without changing the aspect ratio. We assume that there is a good image segmentation algorithm which can detect and crop the object regions from general images, and then the cropped region can be used as the input of the clustering algorithm. For the YCOIL dataset, all images have very simple background, so detection and cropping is not a big problem. A few example images in the original dataset and cropped images are shown in Figure 6.2.

The total number of images in the YCOIL dataset is $18 \times 24 \times 64 = 27648$ and if we consider only illumination subset 2, there are 8208 images. This is too large to be used in the clustering experiment, so we sampled images from the original large dataset. Since images of each object are taken in 24 poses and for each pose there are lighting variations, the algorithm must be able to

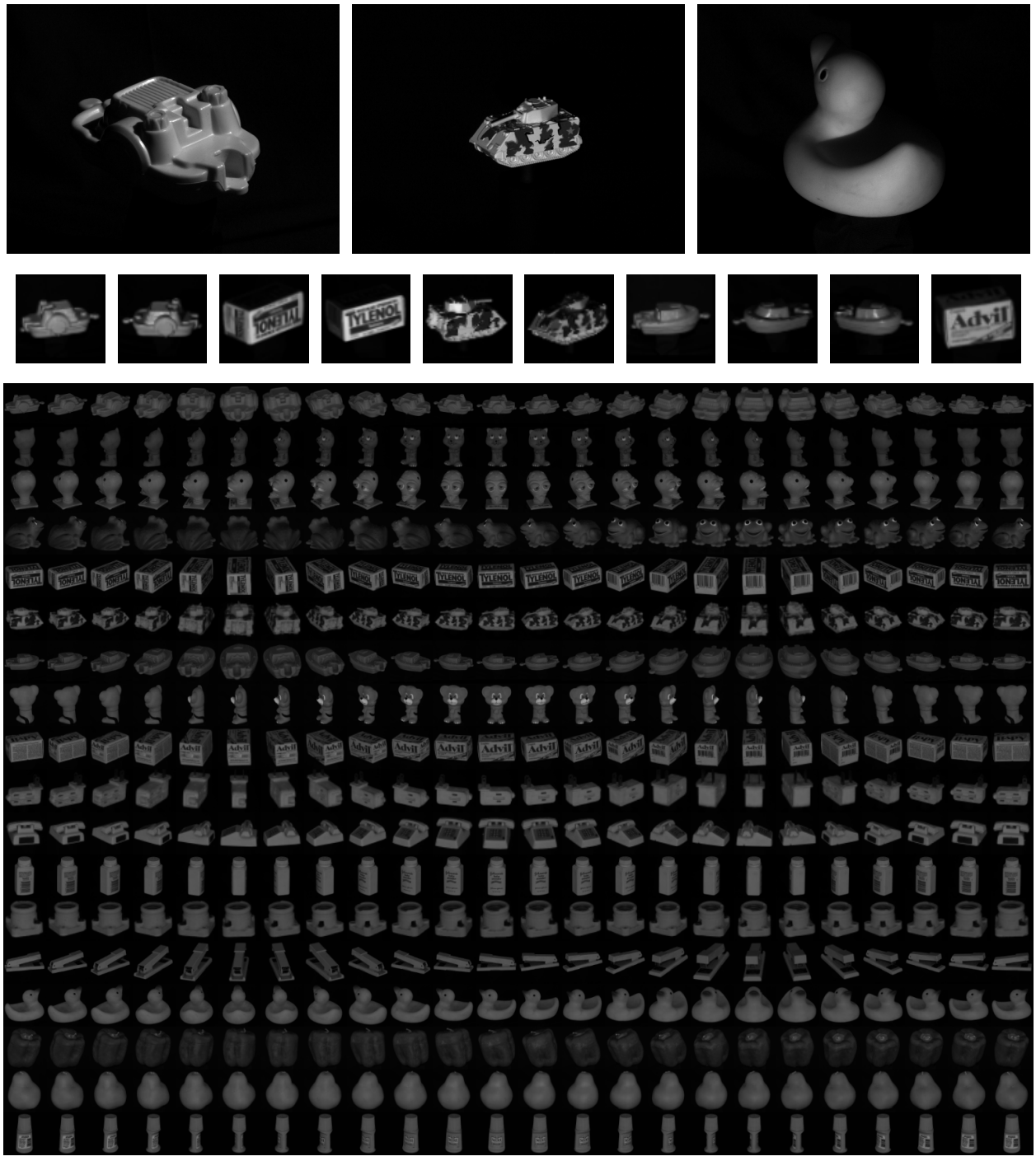
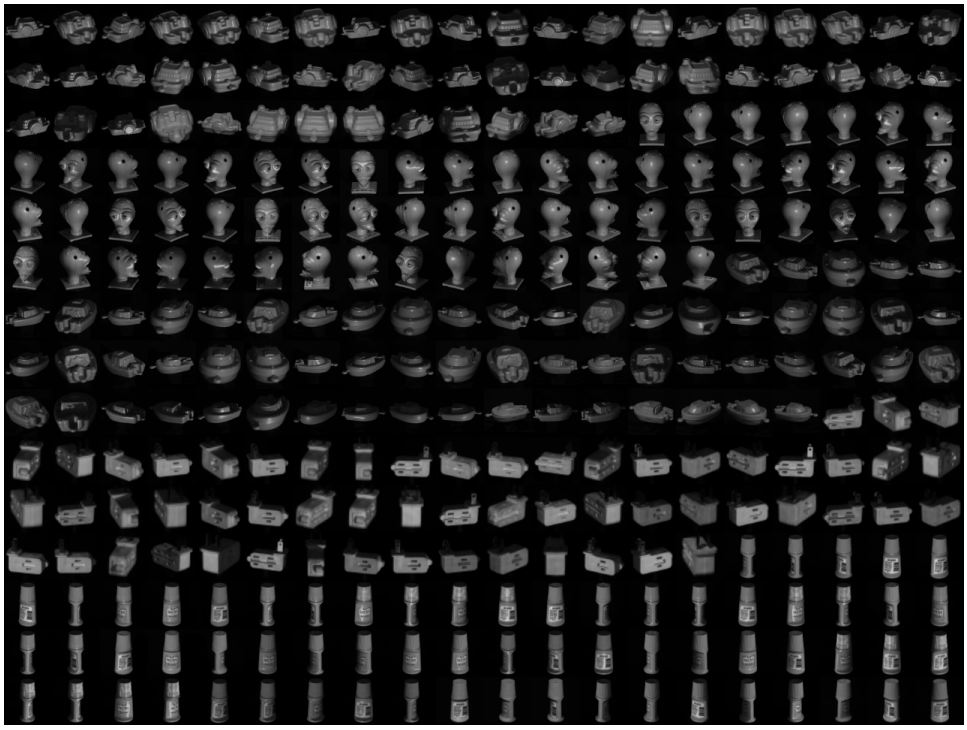


Figure 6.2: Top row: examples of the original YCOIL images. Middle row: examples of the cropped images used in the experiments. Bottom row: simulated ambient images of all poses of all objects.

(a)



(b)

Figure 6.3: (a) one of 10 YCOIL05a datasets without ambient illumination. (b) one of 10 YCOIL05b datasets with simulated ambient light.

find the 'links' among the sampled images of each object. For this task to be doable, there must be enough number of images per objects, or the sampling may not be too sparse. However increasing the sampling rate makes the dataset larger, and this makes the clustering problem difficult. If it is possible to adjust the frequency of sampling, this trade-off must be considered carefully.

We first chose to build datasets containing 300 images of 5 objects. To show how the ambient lighting affects the clustering performance, one dataset is built without simulated ambient lighting (*YCOIL05a*), and the other dataset with the ambient lighting (*YCOIL05b*). Here we did not require that the number of images per object is same, or that all the poses are included in the dataset. However on average, there are 60 images per object, all objects have most poses in the dataset, but again, the lighting may change along with the pose variation. Thus it is not an easy dataset at all.

Next, larger datasets with more objects are built and tested. *YCOIL10* dataset consists of 500 images of 10 randomly selected objects from YCOIL dataset with simulated ambient lighting (just as YCOIL05b). there are fewer images per object (on average 50), and the total number of objects in each dataset is larger. So, these datasets are more difficult to find the correct clustering.

6.2.2 Experimental Results

Table 6.1 and Figure 6.4 shows the clustering results of various algorithms on the dataset YCOIL05a, which has 300 images of 5 objects without simulating ambient illumination. Note that the reported average error rate is not the average of 10 best error rates in the table. Since the best parameters for each testset are different, the numerical average of the best results have little meaning. For each parameter, an average error rate is computed over 10 testsets, and then the best average error rate is picked and shown in the table with the parameters. The best result is obtained by the geodesic extension of the LLS algorithm, and both geo+LLSα and geo+LLSgrad gave exactly same clustering results with geo+LLS algorithm. Generally LGA does not work as well as LLS approaches in this YCOILa experiment. One interesting phenomenon is that the variation in the affine alignment does not affect clustering result at all in this experiment. It is because the regular affine alignment algorithm already finds a good alignment for neighbor images.

Algorithm	LLS(α,grad)	geoLLS(α,grad)	LGA	geoLGA
Testset 1~10	11.00 / 50, 3.10	4.67 / 30, 3.40	7.67 / 20, 2.20	7.67 / 20, 2.20
	4.00 / 50, 1.60	2.00 / 40, 2.30	12.33 / 30, 2.70	12.67 / 30, 2.80
Error (%)	21.67 / 50, 3.10	17.00 / 50, 3.30	29.00 / 05, 1.10	29.00 / 05, 1.10
/ K, $\log_{10}\sigma$	0.00 / 10, 1.50	0.00 / 10, 1.30	5.33 / 05, 0.90	5.33 / 05, 0.90
	34.00 / 50, 3.10	7.33 / 50, 3.20	36.00 / 30, 1.80	36.00 / 30, 1.80
	22.00 / 15, 3.60	13.67 / 20, 3.40	23.67 / 10, 1.70	26.67 / 10, 1.80
	0.67 / 15, 1.70	4.00 / 05, 2.00	6.00 / 20, 2.10	6.00 / 20, 2.10
	12.67 / 20, 2.30	12.00 / 30, 1.60	23.00 / 30, 2.70	22.67 / 30, 2.70
	1.67 / 50, 1.50	1.67 / 50, 2.80	7.67 / 30, 2.80	8.67 / 15, 2.20
	7.00 / 40, 1.40	7.00 / 30, 1.50	20.67 / 10, 1.70	20.00 / 10, 1.70
Average	15.13 / 50, 1.90	12.90 / 50, 2.80	22.50 / 10, 1.90	21.97 / 10, 1.90

Algorithm	LGAcb2	geoLGAcb2	LGAcb3	geoLGAcb3
Testset 1~10	23.00 / 15, 2.40	23.33 / 15, 2.50	24.67 / 30, 3.30	25.33 / 20, 2.80
	17.67 / 15, 2.30	19.00 / 15, 2.40	7.00 / 20, 2.50	11.33 / 40, 3.00
Error (%)	39.33 / 05, 2.50	38.33 / 10, 3.50	29.33 / 50, 3.30	29.00 / 50, 3.30
/ K, $\log_{10}\sigma$	13.67 / 10, 2.20	17.33 / 10, 2.40	15.33 / 30, 3.00	8.00 / 20, 3.40
	23.00 / 10, 1.80	23.00 / 10, 1.80	28.00 / 30, 2.90	26.00 / 30, 3.50
	27.33 / 20, 2.40	27.67 / 15, 2.20	20.33 / 40, 3.20	16.67 / 30, 4.00
	27.00 / 05, 2.10	27.33 / 05, 3.50	19.67 / 10, 2.30	19.67 / 20, 3.10
	36.00 / 15, 2.20	36.00 / 05, 3.00	23.33 / 10, 2.30	21.67 / 10, 2.40
	20.00 / 20, 3.20	20.33 / 10, 2.40	16.33 / 50, 3.40	15.33 / 50, 3.60
	19.33 / 05, 1.50	23.67 / 10, 2.50	10.67 / 15, 2.30	10.33 / 15, 2.50
Average	34.47 / 10, 2.40	32.00 / 10, 2.40	27.50 / 30, 2.90	23.40 / 20, 3.10

Table 6.1: Clustering result of 10 YCOIL05a datasets. The best error rate and its parameters (K,σ) of each testset are reported, and at the end, the best average error rate is also reported. Note that this best average error is not the average of 10 best error rates in each column of the table.

Algorithm	LLS(α,grad)	geoLLS(α,grad)	LGA	geoLGA
Testset 1~10	15.33 / 50, 2.30	9.67 / 50, 3.20	10.67 / 15, 1.30	10.67 / 15, 1.30
	0.33 / 30, 2.10	0.33 / 15, 2.50	1.33 / 20, 1.60	1.33 / 20, 1.60
Error (%)	10.67 / 15, 3.30	13.00 / 05, 3.30	10.00 / 10, 1.70	10.00 / 15, 3.70
/ K, $\log_{10}\sigma$	17.67 / 50, 0.60	16.67 / 15, 2.60	17.33 / 50, 1.80	17.33 / 50, 1.80
	3.33 / 30, 2.30	2.33 / 50, 3.40	0.67 / 20, 1.50	0.67 / 20, 1.50
	0.00 / 05, 2.70	0.00 / 10, 1.10	0.00 / 05, 0.30	0.00 / 05, 0.30
	1.00 / 40, 2.20	1.67 / 50, 2.40	6.00 / 10, 1.70	6.00 / 10, 1.70
	1.00 / 05, 1.90	0.00 / 20, 1.30	2.00 / 15, 0.50	2.00 / 15, 0.50
	0.00 / 50, 2.00	11.00 / 50, 2.10	28.33 / 10, 1.80	24.67 / 10, 1.70
	6.67 / 20, 2.90	5.33 / 30, 0.70	14.33 / 30, 3.00	11.00 / 20, 2.90
Average	11.33 / 20, 3.10	9.57 / 50, 3.20	14.50 / 10, 1.80	14.47 / 10, 1.70

Algorithm	LGAcb2	geoLGAcb2	LGAcb3	geoLGAcb3
Testset 1~10	14.33 / 30, 2.30	14.33 / 30, 2.30	14.67 / 10, 1.90	14.67 / 10, 1.90
	6.00 / 40, 1.60	4.67 / 30, 3.90	0.00 / 20, 1.70	0.00 / 20, 1.70
Error (%)	0.00 / 05, 0.90	0.00 / 05, 0.90	0.00 / 15, 1.30	0.00 / 15, 1.30
/ K, $\log_{10}\sigma$	18.33 / 05, 1.80	15.33 / 05, 0.20	16.67 / 40, 3.40	15.33 / 50, 4.00
	0.00 / 10, 0.70	0.00 / 10, 0.70	0.00 / 05, 1.70	0.00 / 05, 1.70
	0.00 / 05, 0.40	0.00 / 05, 0.30	0.00 / 05, 0.50	0.00 / 05, 0.50
	1.00 / 05, 0.90	1.00 / 05, 0.90	0.00 / 15, 1.50	0.00 / 15, 1.50
	5.67 / 15, 1.20	5.67 / 15, 1.30	7.33 / 05, 2.00	8.33 / 05, 3.90
	11.33 / 40, 1.60	11.33 / 40, 1.60	16.67 / 30, 2.60	15.33 / 15, 2.30
	0.00 / 40, 1.60	0.00 / 30, 1.50	0.00 / 10, 0.70	0.00 / 10, 0.70
Average	10.20 / 05, 3.30	9.93 / 05, 2.90	6.77 / 15, 1.50	6.77 / 15, 1.50

Table 6.2: Clustering result of 10 YCOIL05b datasets with simulated ambient illumination. The best error rate and its parameters (K,σ) of each testset are reported, and at the end, the best average error rate is also reported.

Algorithm	LLS(α,grad)	geoLLS(α,grad)	LGA	geoLGA
Testset 1~16	25.40 / 50, 3.00	20.80 / 30, 2.30	29.80 / 20, 2.60	28.00 / 10, 2.20
	20.60 / 50, 2.10	15.60 / 20, 3.80	29.00 / 30, 2.60	29.00 / 30, 2.60
Error (%)	13.20 / 30, 2.10	10.40 / 40, 3.20	21.00 / 30, 2.70	20.80 / 40, 3.00
/ K, $\log_{10}\sigma$	12.00 / 40, 2.10	9.60 / 50, 3.80	26.60 / 20, 2.50	26.40 / 50, 3.30
	36.60 / 20, 2.20	25.60 / 40, 3.60	42.00 / 40, 3.20	36.60 / 20, 2.40
	25.00 / 50, 1.70	18.40 / 50, 2.00	28.80 / 40, 2.40	28.80 / 40, 2.40
	10.40 / 40, 2.20	9.00 / 40, 2.80	26.40 / 40, 3.10	23.20 / 40, 3.70
	20.40 / 40, 1.80	19.80 / 40, 1.80	30.40 / 05, 3.00	31.80 / 20, 2.60
	19.20 / 50, 1.60	14.40 / 50, 1.90	25.40 / 10, 1.90	24.80 / 10, 1.90
	15.00 / 40, 2.20	11.40 / 30, 2.90	25.60 / 05, 1.60	25.60 / 05, 1.50
	36.60 / 50, 3.40	24.80 / 40, 3.10	39.60 / 20, 2.40	38.40 / 20, 2.40
	16.20 / 10, 3.90	16.00 / 40, 3.40	16.00 / 15, 2.20	14.80 / 20, 2.50
	36.80 / 50, 2.50	31.40 / 15, 3.90	31.80 / 15, 2.10	31.20 / 15, 2.20
	36.00 / 50, 3.50	26.00 / 20, 3.40	32.60 / 30, 3.00	29.40 / 40, 3.30
	16.40 / 50, 1.90	13.20 / 30, 3.50	17.80 / 10, 1.70	17.80 / 10, 1.70
	42.00 / 40, 3.90	32.20 / 20, 3.90	41.80 / 20, 2.70	43.00 / 40, 3.70
Average	29.93 / 50, 2.30	24.36 / 40, 3.10	33.83 / 30, 2.90	32.88 / 20, 2.40

Algorithm	LGAcb2	geoLGAcb2	LGAcb3	geoLGAcb3
Testset 1~16	20.00 / 15, 2.30	15.20 / 15, 2.20	12.20 / 20, 2.40	12.40 / 20, 2.40
	13.00 / 10, 1.40	13.00 / 10, 1.40	16.00 / 10, 2.10	15.40 / 10, 2.40
Error (%)	21.60 / 10, 0.70	17.40 / 20, 3.90	16.60 / 30, 1.40	16.00 / 50, 3.30
/ K, $\log_{10}\sigma$	15.00 / 10, 1.40	15.00 / 10, 1.40	17.40 / 20, 2.10	16.80 / 30, 3.60
	19.40 / 05, 2.30	14.60 / 10, 3.20	15.60 / 20, 2.80	13.00 / 15, 2.70
	12.60 / 15, 1.90	12.40 / 15, 2.40	11.20 / 30, 2.80	11.20 / 30, 2.80
	11.80 / 20, 4.00	4.20 / 05, 3.60	13.00 / 10, 1.90	6.80 / 15, 3.10
	13.20 / 20, 1.20	13.20 / 15, 0.80	8.60 / 10, 1.50	8.60 / 10, 1.50
	10.00 / 10, 2.00	10.20 / 10, 1.80	11.20 / 10, 1.20	11.20 / 10, 1.20
	0.80 / 15, 1.90	0.80 / 15, 1.90	3.60 / 15, 1.90	3.60 / 15, 1.90
	22.20 / 15, 2.10	19.20 / 15, 2.90	8.20 / 10, 1.90	8.00 / 10, 1.90
	17.20 / 05, 1.30	17.20 / 05, 1.30	18.00 / 10, 1.90	17.60 / 15, 2.70
	17.00 / 30, 2.50	10.80 / 10, 4.00	9.00 / 10, 1.90	9.00 / 10, 1.90
	16.60 / 20, 2.40	17.20 / 15, 2.00	12.80 / 30, 2.20	12.80 / 30, 2.20
	14.80 / 05, 1.90	13.40 / 10, 3.30	11.80 / 20, 2.40	11.80 / 20, 2.40
	27.40 / 30, 1.30	25.20 / 15, 4.00	26.60 / 30, 2.70	24.20 / 15, 2.30
Average	19.24 / 15, 2.30	19.70 / 15, 2.20	17.99 / 20, 2.40	17.81 / 20, 2.40

Table 6.3: Clustering result of 16 YCOIL10 datasets. The best error rate and its parameters (K,σ) of each testset are reported, and at the end, the best average error rate is also reported.

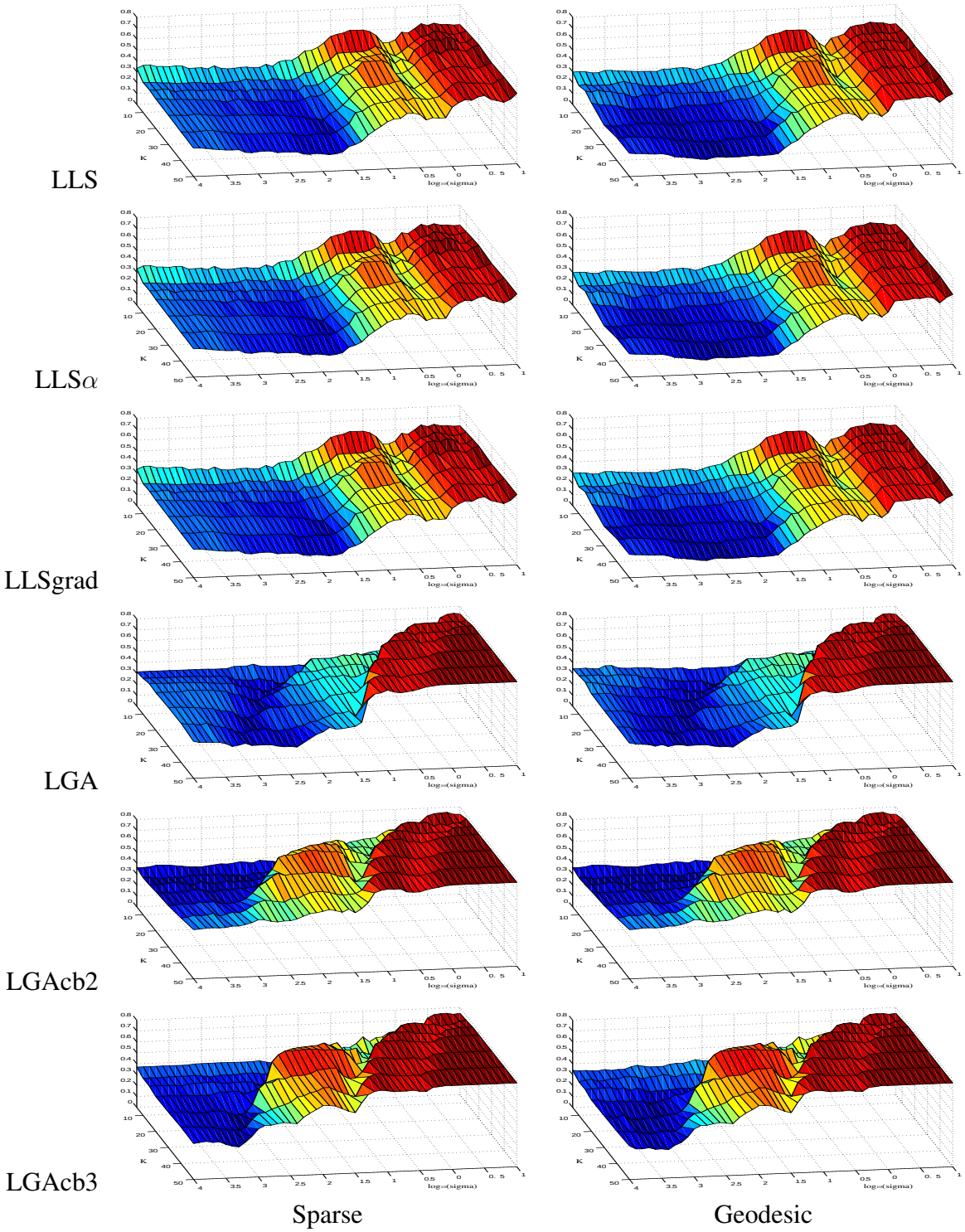


Figure 6.4: Average clustering results of 10 YCOIL05a datasets.

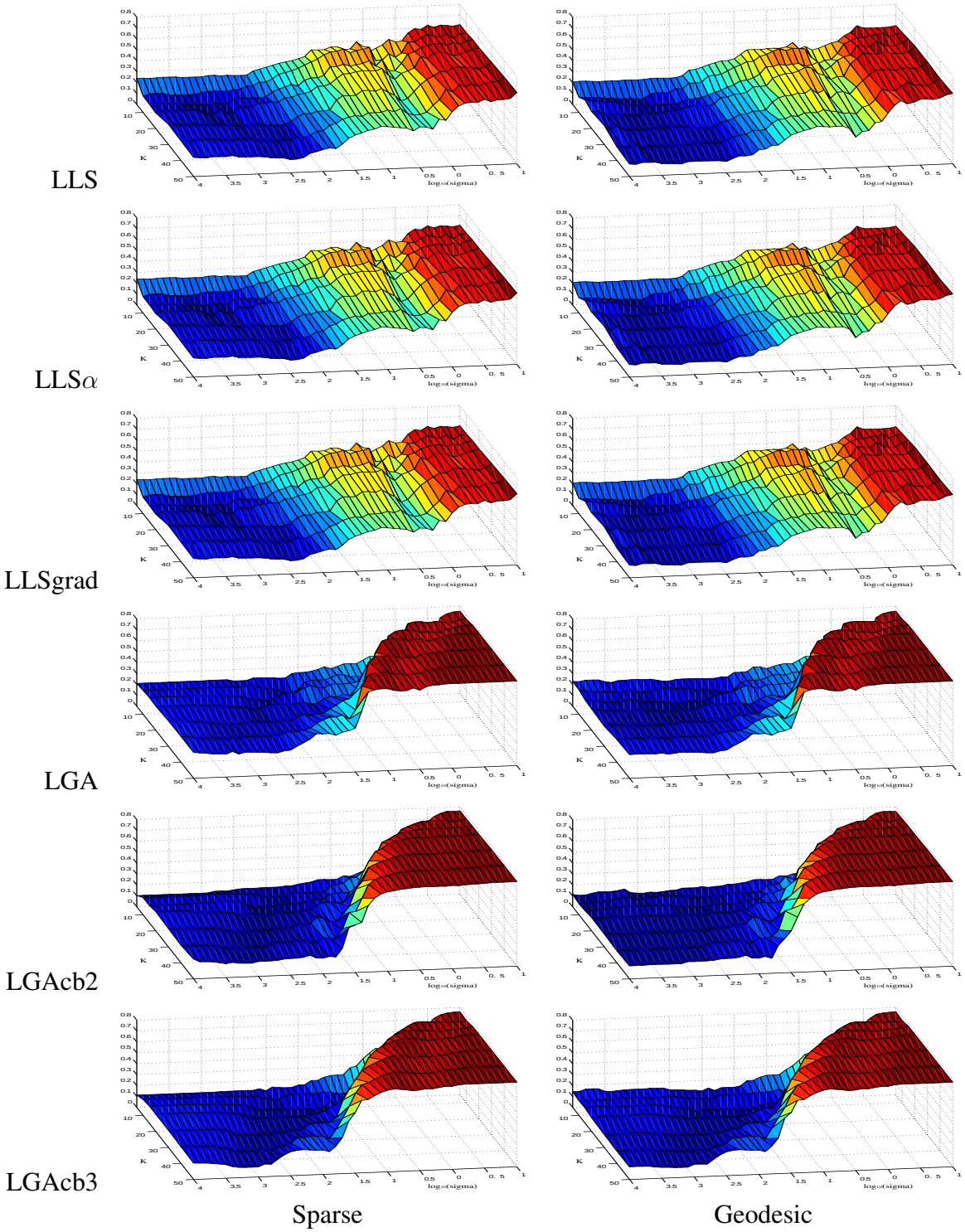


Figure 6.5: Average clustering results of 10 YCOIL05b datasets.

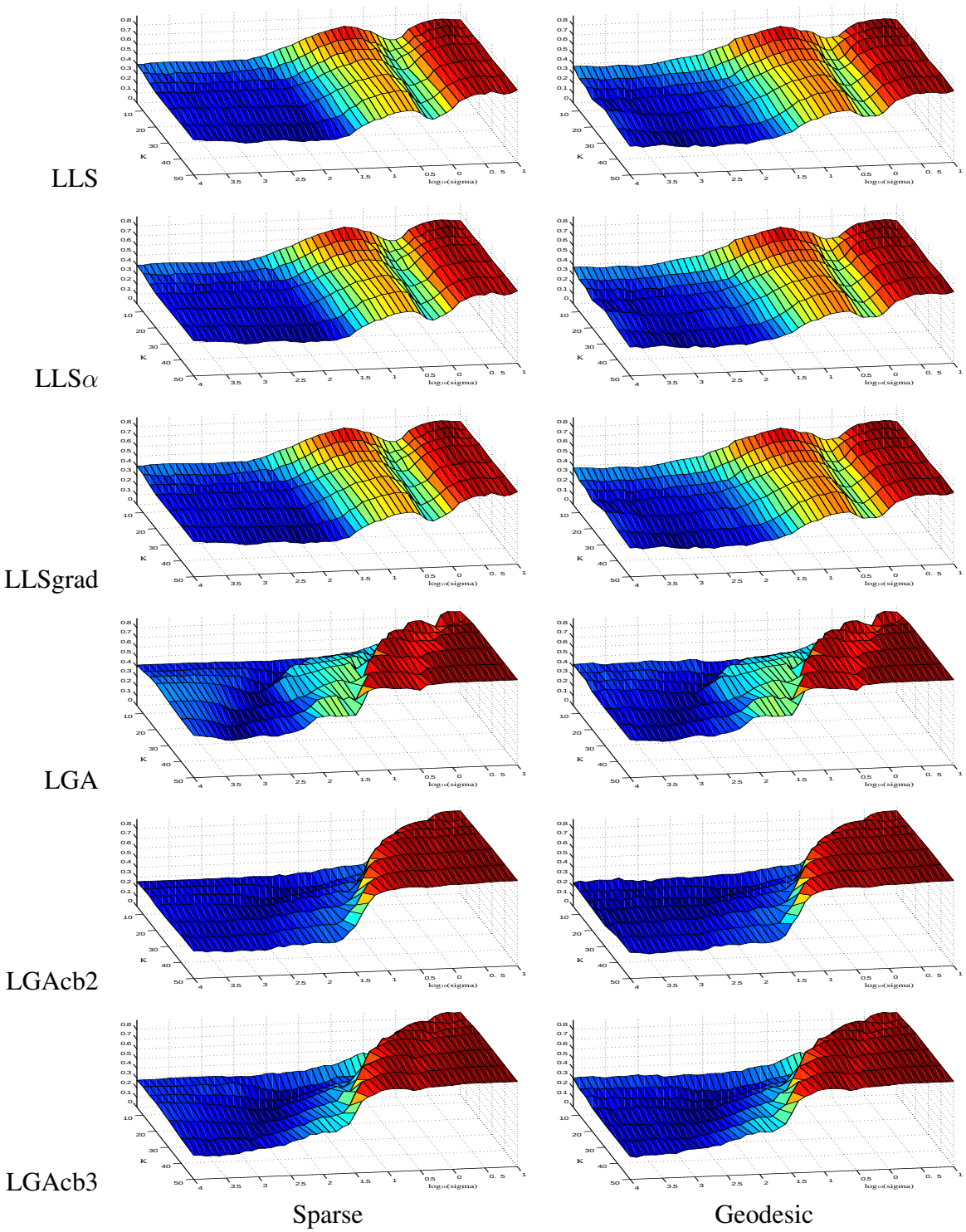


Figure 6.6: Average clustering results of 16 YCOIL10 datasets.

Algorithm	YCOIL05a	YCOIL05b	YCOIL10
LLS / LLSα / LLSgrad	15.13 / 50, 1.90	11.33 / 20, 3.10	29.93 / 50, 2.30
geo+ LLS / LLSα / LLSgrad	12.90 / 50, 2.80	9.57 / 50, 3.20	24.36 / 40, 3.10
LGA	22.50 / 10, 1.90	14.50 / 10, 1.80	33.83 / 30, 2.90
geo+LGA	21.97 / 10, 1.90	14.47 / 10, 1.70	32.88 / 20, 2.40
LGAcb2	34.47 / 10, 2.40	10.20 / 05, 3.30	19.24 / 15, 2.30
geo+LGAcb2	32.00 / 10, 2.40	9.93 / 05, 2.90	19.70 / 15, 2.20
LGAcb3	27.50 / 30, 2.90	6.77 / 15, 1.50	17.99 / 20, 2.40
geo+LGAcb3	23.40 / 20, 3.10	6.77 / 15, 1.50	17.81 / 20, 2.40

Table 6.4: Best average error rates of proposed algorithms

Compared to the YCOIL05a results, all algorithms perform better on the YCOIL05b datasets (see Table 6.2 and Figure 6.5). It obviously shows that the existence of ambient lighting helps in clustering since it gives better idea how the object looks like. Especially for the gradient affinity based algorithm, the enhencement is quite large due to the additional information from the regions that are shadowed in the YCOIL05a setup. One can also notice that the bump around the mid-range of σ in the parameter scan graph has disappeared.

Results of YCOIL10 datasets (Figure 6.6 and Table 6.3) do not present very good performance, and this is not surprising since YCOIL10 datasets are designed to be harder than the two previous datasets. Here we can also verify the importance of ambient lighting from the fact that the experimental results are qqualitatively more similar to YCOIL05b than YCOIL05a (the gradient based affinity performs better than LLS based ones).

Table 6.4 summarizes the results reported in this section. When there is ambient illumination, 3D histogram based gradient affinity seems to work best. One prominent and important thing in these experiments is that the geodesic expansion improves the clustering results for most of the experiments. Although the conversion function is picked rather arbitrarily, this experimental result shows it improves the accuracy and robustness of the clustering algorithms regardless of the type of affinity measure they use.

6.3 Discussion

In this chapter, we presented several extensions of the previous illumination-only or pose-only clustering algorithms. The key issues are to achieve accurate and robust alignment of neighboring images to the target image, and to compute the affinity correctly from the roughly aligned images.

From the experiments, geodesic expansion of the LLS affinity matrix seems to perform best in many cases. We can also claim that in general non-negative weight estimation algorithm works better than the pairwise gradient image comparision methods. As future work, the reason for the superior performance of geodesic expansion needs to be investigated. So far, none of the proposed algorithm gives very compelling performance, there is a clear need for additional study to find better techniques.

Chapter 7

Conclusion and Future Work

In this thesis, a couple of interesting problems are addressed and studied. Besides the well-studied graph partitioning algorithms or spectral clustering algorithm, we extended the domain from pairwise similarities to multi-adic relations, and provided an effective algorithm that finds a graph approximation of the given hypergraph. Compared to the existing hypergraph algorithms, we prove and show empirical result that the proposed algorithm performs robustly better.

The main theme in this thesis is clustering images of objects according to their identity. There are many other factors that each image contains, like illumination, viewing direction, etc., and the goal of our work is to identify and separate those factors so that the affinity between images effectively represents the identity relation.

The first hurdle is varying illumination. There has been a few solid theory about how illumination affects the appearance of objects, and we used two of them. One is the illumination cone theorem, and the conic affinity is defined based on that theorem. It assumes the object surface is Lambertian and piecewisely planar, and tries to find the generators of the cone. This process can be written as computing the non-negative weights for other images in the dataset, and the weights shows the proximity of the image to the target image. The gradient affinity is from a study on the properties of gradients images of a same object. Even when lighting changes, the image gradient vector does not change much, and the change also can be modeled in a probabilistic framework. The gradient affinity measure uses a simplified version of this probabilistic measure and presents a good performance in clustering.

The more difficult-to-handle factor is the viewing direction change. When object moves or viewing direction changes, the image of the object changes in a very complecated way in the image space (often called as a image manifold). To handle the pose changes, we concentrate to the nearby images and try to find how likely a neighbor image is from the same object. Again non-negative coefficients of each image which gives minimal least-square error can be effectively used as a measure of closeness. The fact that an affine warp can give good approximation of the apperance change when the 3D motion is small leads us to try affine alignments before compare the neighbor images. With these two components, LLS gave very impressive clustering results on various datasets with pose variations.

Most of the time in real world, both illumination and pose changes together, so the clustering algorithm needs to handle the simultaneous changes. We proposed a few possible directions, and some performed well, but the problem still remains open and needs to be studied. Geodesic expansion of a sparse affinity matrix often improves the clustering result and makes the algorithm more robust empirically, but still theoretic explanation is missing. So far, all affinity measures and clustering algorithm treats the image as a whole. It worked out well for two earlier problems, but we also think part-based or feature-based approach may give better result, and currently we are seeking in that direction.

To be a useful algorithm, it must be simple to use and show robust results over parameter selection. In all algorithms propsed here, the number of parameters is not many, and the experiments show that close to the best performance can be achieved in a relatively large range of parameter values. Like other clustering literature, we can only show the robust performance empirically by experiments. One very interesting future work is to develop a measure of robustness of a clustering algorithm on parameter selection.

References

[1] Y. Adini, Y. Moses, and S. Ullman. Face recognition: The problem of compensating for changes in illumination direction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):721–732, 1997.

[2] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[3] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19(1–2):1–81, 1995.

[4] F. R. Bach and M. I. Jordan. Learning spectral clustering. Technical Report UCB/CSD-03-1249, University of California Berkeley, June 2003.

[5] F. R. Bach and M. I. Jordan. Learning spectral clustering. In *Advances in Neural Information Processing Systems*, 16, 2004.

[6] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *Int'l Journal of Computer Vision*, 56(3):221–255, March 2004.

[7] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[8] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *Proc. Int'l Conf. on Computer Vision*, volume 2, pages 383–390, 2001.

[9] R. Basri, D. Roth, and D. Jacobs. Clustering appearances of 3D objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 414–420, 1998.

[10] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions. In *Int'l Journal of Computer Vision*, volume 28, pages 245–260, 1998.

[11] I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32(1):29–73, 1985.

[12] T. O. Binford. Visual perception by computer. In *Proc. IEEE Conf. on Systems and Control*, 1971.

[13] A. Björck. *Numerical methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[14] I. Borf and P. Groenen. *Modern multidimensional scaling: Theory and Applicatsions*. Springer Series in Statistics. Springer Verlag, 1997.

[15] R. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17(1-3):285–349, 1981.

[16] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckert-Young decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[17] H. Chen, P. Belhumeur, and D. Jacobs. In search of illumination invariants. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 254–261, 2000.

[18] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[19] T. Cox, M. Cox, and J. Branco. Multidimensional scaling for n-tuples. *British Journal of Mathematical and Statistical Psychology*, 44:195–206, 1991.

[20] M.-M. Deza and I. Rosenberg. n-Semimetrics. *European Journal of Combinatorics*, 21:797–806, 2000.

[21] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical report, UTCS Technical Report #TR-04-25, July 2004.

[22] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, Secound Edition*. John Wiley and Sons, 2001.

[23] R. Epstein, P. Hallinan, and A. Yuille. 5+/-2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *IEEE Workshop on Physics-Based Modeling in Computer Vision*, 1995.

[24] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th conference on Design automation*, pages 175–181. IEEE Press, 1982.

[25] A. W. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proc. European Conf. on Computer Vision*, LNCS 2353, pages 304–320. Springer-Verlag, 2002.

[26] A. W. Fitzgibbon and A. Zisserman. Joint manifold distance: a new approach to appearance based clustering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 26–33, 2003.

[27] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[28] B. Frey and N. Jojic. Fast, large-scale transformation-invariant clustering. In *Advances in Neural Information Processing Systems 14*, pages 721–727, 2001.

[29] H. Frigui, N. Boujemaa, and S. Lim. Unsupervised clustering and feature discrimination with application to image database categorization. In *Joint 9th IFSA World Congress and 20th NAFIPS Conference*, 2001.

[30] Y. Gdalyahu, D. Weinshall, and M. Werman. Stochastic image segmentation by typical cuts. In *CVPR*, 1999.

[31] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Generative models for recognition under variable pose and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 40(6):643–660, 2001.

[32] D. Gibson, J. M. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 311–322. Morgan Kaufmann Publishers Inc., 1998.

[33] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1997.

[34] J. Goldberger, H. Greenspan, and S. Gordon. Unsupervised image clustering using the information bottleneck method. In *DAGM Symposium*, pages 158–165, 2002.

[35] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *CVPR (1)*, pages 1150–1157, 2005.

[36] D. B. Graham and N. M. Allinson. Norm^2-based face recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 586–591, 1999.

[37] S. W. Hadley. Approximation techniques for hypergraph partitioning problems. *Discrete Appl. Math.*, 59(2):115–127, 1995.

[38] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[39] C. Hayashi. Two dimensional quantification based on the measure of dissimilarity among three elements. *Annals of the Institute of Statistical Mathematics*, 24:251–257, 1972.

[40] W. J. Heiser and M. Bennani. Triadic distance models: Axiomatization and least squares representation. *Journal of Mathematical Psychology*, 41:189–206, 1997.

[41] D. J. Higham and M. Kibble. A unified view of spectral clustering. Technical Report 2, University of Strathclyde, Department of Mathematics, University of Strathclyde, Glasgow G1 1XH, UK, January 2004.

[42] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearance of objects under varying lighting conditions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 11–18, 2003.

[43] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[44] T. Hu and K. Morder. Multiterimnal flows in hypergraphs. In T. Hu and Ku, editors, *VLSI Circuit Layout: Theory and Design*, pages 87–93. IEEE Press, 1985.

[45] E. Ihler, D. Wagner, and F. Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Information Processing Letters*, 45:171–175, 1993.

[46] D. W. Jacobs, P. N. Belhumeur, and R. Basri. Comparing images under varying illumination. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 610–617. IEEE, IEEE Press, 1998.

[47] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[48] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *P-IEEE*, 80:1502–1517, 1992.

[49] S. Joly and G. L. Calvé. Three-way distances. *Journal of Classification*, 12:191–205, 1995.

[50] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 343–348. ACM Press, 1999.

[51] L. Kaufman and P. J. Rousseeuw. *Finding groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[52] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, 49:291–307, February 1970.

[53] J. Keuchel, C. Schnörr, C. Schellewald, and D. Cremers. Binary partitioning, perceptual grouping, and restoration with semidefinite programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1364–1379, November 2003.

[54] J. Kleinberg. An impossibility theorem for clustering. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 446–453. MIT Press, Cambridge, MA, 2003.

[55] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:781–791, 1999.

[56] K.-C. Lee, J. Ho, and D. Kriegman. Nine points of lights: Acquiring subspaces for face recognition under variable lighting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 519–526, 2001.

[57] F. Li, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Proc. Int'l Conf. on Computer Vision*, volume 2, page 1134, 2003.

[58] S. Z. Li, X. Lv, and H. Zhang. View-based clustering of object appearances based on independent subspace analysis. In *Proc. Int'l Conf. on Computer Vision*, pages 295–300, 2001.

[59] J. Lim, J. Ho, M.-H. Yang, K.-C. Lee, and D. Kriegman. Image clustering with metric, local linear structure and affinity symmetry. In *Proc. European Conf. on Computer Vision*, volume 1, pages 456–468, 2004.

[60] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l Journal of Computer Vision*, 14(1):5–24, 1995.

[61] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. Ditterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 15*, pages 849–856. MIT Press, 2002.

[62] A. P. Pentland. Finding the illuminant direction. *J. of Optical Society America A*, 72(4):448–455, 1982.

[63] P. Perona and M. Polito. Grouping and dimensionality reduction by locally linear embedding. In *Advances in Neural Information Processing Systems 15*, pages 1255–1262, 2002.

[64] J. Pistorius and M. Minoux. An improved direct labeling method for the max-flow min-cut computation in large hypergraphs and applicatsions. *International Transactions in Operational Research*, 10(1):1–11, 2003.

[65] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Proc. SIGGRAPH*, pages 117–228, 2001.

[66] B. Raytchev and H. Murase. Unsupervised face recognition from image sequences based on clustering with attraction and repulsion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 25–30, 2001.

[67] B. Raytchev and H. Murase. Unsupervised recognition of multi-view face sequences based on pairwise clustering with attraction and repulsion. *Computer Vision and Image Understanding*, 91(1/2):22–52, 2003.

[68] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[69] S. T. Roweis and l. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[70] B. L. Saux and N. Boujemaa. Unsupervised robust clustering for image database categorization. In *Proc. Int'l Conf. on Pattern Recognition*, volume 1, pages 259–262, 2002.

[71] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[72] A. Shashua. Geometry and photometry in 3d visual recognition. *Ph.D. Thesis, MIT AITR-1401*, Nov 1992.

[73] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proc. Int'l Conf. on Computer Vision*, pages 1154–1160, 1998.

[74] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[75] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002.

[76] T. Sim and T. Kanade. Combining models and exemplars for face recognition: An illuminating example. In *CVPR workshop on Models versus Exemplars in Computer Vision*, 2001.

[77] P. Simard, Y. L. Cun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition - tangent distance and tangent propagation. In *Neural Networks*, volume 1524, pages 239–274, 1998.

[78] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, pages 2319–2323, December 2000.

[79] P. H. S. Torr. Geometric motion segmentation and model selection. In J. Lasenby, A. Zisserman, R. Cipolla, and H. Longuet-Higgins, editors, *Philosophical Transactions of the Royal Society A*, pages 1321–1340. Roy Soc, 1998.

[80] G. T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recogn*, 12(4):261–268, 1980.

[81] S. Ullman. On visual detection of light sources. *Biological Cybernetics*, 21:205–212, 1976.

[82] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems*, 2005.

[83] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proc. European Conf. on Computer Vision*, pages 18–32, 2000.

[84] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proc. Int'l Conf. on Computer Vision*, volume 2, pages 975–982, 1999.

[85] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1101–1113, November 1993.

[86] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. Int'l Conf. on Computer Vision*, pages 11–17, 2003.

[87] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.

[88] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

[89] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):680–702, 1991.

[90] J. Zien, M. Schlag, and P. Chan. Multi-level spectral hypergraph partitioning with arbitrary vertex sizes. In *Proc. ICCAD*, pages 201–204, 1996.